VISIT…

LANZAROTE
Caliente.COM

БИБЛИОТЕКА ПРОГРАММИСТА



Оливер Монтенбрук
Томас Пфлегер

Астрономия на персональном компьютере

Четвертое издание

ПИТЕР

Оливер Монтенбрук
Томас Пфлегер

Астрономия на персональном компьютере

Четвертое издание

Эта книга не только заменит вам астрономический календарь и астрономический ежегодник, но и даст основные знания из области небесной механики и методов обработки наблюдений. Вы найдете в ней всю необходимую информацию и готовые программные решения.

Прочитав эту книгу, вы сможете самостоятельно вычислять:

* положения Солнца, Луны и планет;
* моменты восхода и захода;
* физические эфемериды Солнца и больших планет;
* положения комет и малых планет (с учетом пертурбаций);
* фазы Луны;
* полосы полной фазы и местные обстоятельства солнечных затмений;
* покрытия звезд Луной;
* орбиты небесных тел на основании трех наблюдений;
* координаты объектов по фотографическим изображениям.



Компакт-диск содержит исходные тексты всех описанных в книге программ, откомпилированные версии программ для исполнения в среде как Windows 9x/NT, так и Linux, а также три обширные базы данных: каталог положений и собственных движений более чем 400 тыс. звезд, зодиакальный каталог и каталог элементов орбит всех открытых на момент написания книги малых планет.

Требования к системе:

* Windows 95/98/ME/NT/2000/XP или Linux с ядром 2.x или более новым;
* процессор i486 или более мощный;
* 8 Мбайт оперативной памяти;
* 10 Мбайт свободного дискового пространства (рекомендуется 150 Мбайт);
* не обязательно, но желательно иметь установленный компилятор Microsoft Visual C++ или GNU C++.

ПИТЕР
WWW.PITER.COM

ISBN 5-318-00223-4



Посетите наш web-магазин: **www.piter.com**

СЕРИЯ

БИБЛИОТЕКА ПРОГРАММИСТА



Oliver Montenbruck, Thomas Pfleger

Astronomy on the Personal Computer

Translated by Storm Dunlop
With a Foreword by Richard M. West
Fourth, Completely Revised Edition
With 46 Figures and CD-ROM

Оливер Монтенбрук, Томас Пфлегер

БИБЛИОТЕКА ПРОГРАММИСТА

Астрономия на персональном компьютере

Четвертое издание

Санкт-Петербург
Москва • Харьков • Минск
2002

ПИТЕР®

Оливер Монтенбрук, Томас Пфлегер

Астрономия на персональном компьютере

Перевели с английского А. Сергеев, А. Телов

Главный редактор	*Е. Строганова*
Заведующий редакцией	*И. Корнеев*
Руководитель проекта	*А. Пасечник*
Научный редактор	*А. Пасечник*
Художник	*Н. Биржаков*
Иллюстрации	*П Быстров*
Корректор	*В. Листова*
Верстка	*П. Быстров*

ББК 22.6с11+32.973.23
УДК 52:681.3

Монтенбрук О., Пфлегер Т.

М77 Астрономия на персональном компьютере (+CD). — СПб.: Питер, 2002. — 320 с.: ил.

ISBN 5-318-00223-4

Эта книга не только заменит вам астрономический календарь и астрономический ежегодник, но и даст основные знания из области небесной механики и методов обработки наблюдений. Вы найдете в ней всю необходимую информацию и готовые программные решения для предвычисления положений Солнца, Луны и планет, восходов и заходов светил, физических эфемерид Солнца и больших планет, положений комет и астероидов, фаз Луны, обстоятельств солнечных и лунных затмений, покрытий звезд Луной, расчета элементов орбиты по трем наблюдениям и многое другое.

Прилагаемый компакт-диск содержит исходные тексты всех описанных в книге программ, откомпилированные версии программ для исполнения в среде как Windows 9x/NT, так и Linux, а также три обширные базы данных: каталог положений и собственных движений более чем 400 тыс. звезд, зодиакальный каталог и каталог элементов орбит всех открытых на момент написания книги малых планет.

Для астрономов, астрологов, любителей и профессионалов, а также для всех, интересующихся астрономическими наблюдениями.

Original English language Edition Copyright © Springer-Verlag Berlin Heidelberg 1991,1994, 1998, 2000

© Перевод на русский язык, А. Сергеев, А. Телов, 2002

© Издательский дом «Питер», 2002

Права на издание получены по соглашению с Springer-Verlag

Все права защищены Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственность за возможные ошибки, связанные с использованием книги

ISBN 5-318-00223-4
ISBN 3-540-67221-4 (англ.)

ЗАО «Питер Бук». 196105, Санкт-Петербург, Благодатная ул., д. 67.
Лицензия ИД № 01940 от 05.06.00.

Налоговая льгота – общероссийский классификатор продукции ОК 005-93, том 2; 953000 – книги и брошюры.

Подписано в печать 22 11.01. Формат 70×100/16. Усл. п. л 25,8 Тираж 3000 экз. Заказ № 2276.

Отпечатано с готовых диапозитивов в ФГУП «Печатный двор» им. А. М. Горького
Министерства РФ по делам печати, телерадиовещания и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.

Содержание

Предисловия **9**

От издательства 15

1. Введение **16**

1.1. Несколько примеров 16
1.2. Астрономия и вычисления 17
1.3. Приемы и языки программирования 18

2. Системы координат **22**

2.1. Приступаем 22
2.2. Календарь и юлианские дни 29
2.3. Эклиптические и экваториальные координаты 31
2.4. Прецессия 34
2.5. Геоцентрические координаты и солнечная орбита 39
2.6. Программа Coco 42

3. Вычисление моментов восхода и захода **49**

3.1. Горизонтальная система координат наблюдателя 49
3.2. Солнце и Луна 52
3.3. Звездное время и часовой угол 53
3.4. Всемирное и эфемеридное время 55
3.5. Параллакс и рефракция 58
3.6. Моменты восхода и захода 60
3.7. Квадратичная интерполяция 62
3.8. Программа Sunset 63
3.9. Программа Planrise 70

4. Кометные орбиты **72**

4.1. Форма и ориентация кометной орбиты 72
4.2. Положение на орбите 74
4.3. Численное решение уравнения Кеплера 78
4.4. Околопараболические орбиты 82
4.5. Векторы Гаусса 85
4.6. Световая задержка 89
4.7. Программа Comet 90

5. Возмущения орбит . . . 96
5.1. Уравнение движения . . . 97
5.2. Планетные координаты . . . 99
5.3. Численное интегрирование . . . 102
5.4. Оскулирующие элементы . . . 106
5.5. Программа Numint . . . 108
5.6. База данных элементов орбит астероидов . . . 116

6. Планетные орбиты . . . 118
6.1. Разложение в ряд уравнения Кеплера . . . 119
6.2. Возмущающие члены . . . 122
6.3. Численное суммирование рядов . . . 125
6.4. Видимые и астрометрические координаты . . . 131
6.4.1. Аберрация и световая задержка . . . 131
6.4.2. Нутация . . . 134
6.5. Программа Planpos . . . 136

7. Физические эфемериды планет . . . 141
7.1. Вращение . . . 141
7.1.1. Позиционный угол оси вращения . . . 142
7.1.2. Планетографические координаты . . . 144
7.2. Условия освещения . . . 151
7.2.1. Фаза и элонгация . . . 151
7.2.2. Позиционный угол Солнца . . . 153
7.2.3. Видимая звездная величина . . . 154
7.2.4. Диаметр видимого диска . . . 155
7.3. Программа Phys . . . 155

8. Орбита Луны . . . 160
8.1. Общие свойства лунной орбиты . . . 160
8.2. Теория Брауна . . . 164
8.3. Приближение Чебышева . . . 175
8.4. Программа Luna . . . 181

9. Солнечные затмения . . . 185
9.1. Фазы Луны и затмения . . . 185
9.2. Геометрия затмений . . . 188
9.3. Географические координаты и сплюснутость Земли . . . 191
9.4. Продолжительность затмения . . . 195
9.5. Координаты Солнца и Луны . . . 196
9.6. Программа Eclipse . . . 197
9.7. Обстоятельства затмений . . . 206
9.8. Программа Ecltimer . . . 208

10. Покрытия звезд Луной 211
10.1. Видимые положения 212
10.2. Геоцентрическое соединение 216
10.3. Главная плоскость 220
10.4. Покрытия и открытия 223
10.5. Программа Occult 225
10.6. Оценка из наблюдений величины ΔT=ET−UT 235

11. Вычисление орбит 237
11.1. Вычисление орбиты по двум позиционным векторам 237
11.1.1. Отношение площадей сектора и треугольника 238
11.1.2. Элементы орбиты. 241
11.2. Упрощенный метод Гаусса 246
11.2.1. Геометрия наблюдений, отнесенных к центру Земли 246
11.2.2. Последовательное приближение для отношения площадей сектора и треугольника 249
11.2.3. Множественные решения 250
11.3. Полный метод Гаусса 251
11.3.1. Уравнение Гаусса—Лагранжа 251
11.3.2. Улучшенная итерация для отношения площадей треугольников. . 254
11.3.3. Время распространения света 254
11.4. Программа Gauss 255

12. Астрометрия 266
12.1. Фотографическое изображение 266
12.2. Постоянные пластинки 269
12.3. Метод наименьших квадратов 271
12.4. Программа Foto 275
12.5. Каталог PPM (Position and Proper Motion Catalogue) 281

Приложение 284
П.1. Прилагаемый компакт-диск 284
П.1.1. Содержание. 284
П.1.2. Требования к системе 285
П.1.3. Запуск программ 286
П.2. Компиляция и сборка программ 287
П.2.1. Общие рекомендации по адаптации к компилятору 287
П.2.2. Microsoft Visual C++ for Windows 95/98/NT. 288
П.2.3. GNU C++ for Linux 289
П.3. Список библиотечных функций. 290

Список обозначений 295

Словарь терминов. 299

Литература **304**
Общие работы 304
Глава 2 306
Глава 3 306
Глава 4 306
Глава 5 307
Глава 6 308
Глава 7 309
Глава 9 310
Глава 10 310
Глава 11 311
Глава 12 311
Алфавитный указатель **313**

Предисловие

Астрономия — древнейшая наука, уходящая своими корнями в глубокое прошлое. И вместе с тем, это одна из самых современных и быстро развивающихся областей фундаментального знания, которая исследует вопросы, имеющие огромное значение для человечества. Занимаясь на протяжении тысячелетий преимущественно описанием наблюдений, в последнее время астрономия стала высокотехнологичной сферой деятельности, стремящейся к глубокому пониманию окружающего нас космоса и, что еще важнее, нашего места в нем. Современная астрономия — это чрезвычайно многогранная наука, крепко связанная со многими другими научными направлениями: физикой и химией, метеорологией и минералогией, оптикой, механикой, электроникой, и конечно, с высшей математикой и вычислительными методами. Важное место в этом списке должна занимать и философия.

В последние годы темпы развития астрономии стали просто потрясающими. Это касается как наблюдательной, так и вычислительной техники. Используя все более крупные телескопы, все лучшие инструменты, профессиональные наблюдатели стараются заглянуть в неизведанное, лежащее за горизонтами современного знания. Все более быстрые и надежные каналы связи и всепроникающая глобальная сеть Интернет открывают возможности для совершенно нового уровня международного сотрудничества. Но особенно впечатляют скорость и целеустремленность, с которыми любители астрономии реагируют на появление всех этих новых возможностей. Улучшение качества и доступности новых любительских инструментов, в особенности появление относительно недорогих CCD-камер, позволяет современным любителям вторгаться в такие области, которые еще недавно были доступны исключительно их профессиональным коллегам.

Эта книга — тоже является отчетливым признаком идущего развития. С момента появления ее первого немецкого издания 10 лет назад, она стала настольной книгой для серьезных любителей астрономии. Вышедшее спустя год первое английское издание сделало ее доступной постоянно растущему мировому любительскому сообществу. Но не секрет, что эта книга оказалась полезной далеко не только любителям. Немало профессионалов — и я в их числе — узнали из нее немало важных и интересных подробностей.

Почему эта книга оказалась настолько важной? На этот вопрос можно ответить по-разному. Каждый читатель, вероятно, найдет свои причины. Одному важно на много лет вперед предвычислять лунные фазы и солнечные затмения, другому — точно определять положения астероидов по соседним звездам, входящим в астрометрические стандарты. Конкретных мотивов может быть много, но я уверен, что есть другие более важные и более общие причины.

Рассмотренные в книге темы и методы составляют фундаментальную и абсолютно необходимую часть астрономии. Без умения точно определять моменты небесных явлений, без знания, как вычислить положения планет, комет и звезд для заданного места и времени, наблюдательная астрономия, в том виде, как мы ее знаем сегодня, просто невозможна.

Читатели найдут в книге простые и точные объяснения, как произвести нужные вычисления, а также описание программных средств, обеспечивающих высокую точность расчетов. Особенно важна возможность посмотреть и при необходимости адаптировать для своих целей исходный код программ, которые теперь представлены на языке C++. Таких возможностей вам не предоставит ни одна из популярных программ-планетариев, которые предназначены для менее требовательных пользователей. Размещенный на прилагаемом диске обширный каталог объектов содержит данные, необходимые для работы ряда программ. Дополнительные возможности обеспечивает web-страница, на которой размещаются обновления и полезные ссылки.

Книга позволяет читателю-пользователю вступить на «профессиональную» территорию. Она дает нечто большее, чем простое удовлетворение технических потребностей астрономических наблюдений. Она дарит вам радость работы с тщательно продуманным программным обеспечением на своем домашнем компьютере и удовлетворение от все более глубокого проникновения в механику природы. И плюс ко всему этому она позволяет серьезным любителям получать результаты профессионального качества, что дает им возможность сделать свой вклад в развитие астрономии.

Авторов следует поздравить с той выдающейся работой, которую они осуществили на благо астрономического сообщества. Я не сомневаюсь, что эта замечательная книга будет долго оставаться одним из наиболее популярных руководств во всех астрономических клубах и у всех, кто регулярно занимается астрономическими наблюдениями.

Ричард М. Вест,
Европейская южная обсерватория
Гарчинг
Ноябрь 1999

Предисловие авторов к четвертому изданию

Оглянувшись на прошедшие десять лет, мы с большим облегчением обнаружили, что астрономические и математические процедуры, изложенные в книге «Астрономия на персональном компьютере», и к концу тысячелетия остаются практически во всем правильными и точными. Огромное впечатление на нас производит стремительное, скорее даже взрывное, развитие компьютерной техники. В результате во многих домах компьютер занял теперь прочное место рядом с телефоном и телевизионным приемником.

Поэтому не должно удивлять, что в этом новом издании мы в основном сконцентрировались на фундаментальном пересмотре и модернизации всех программ. На фоне постепенного упадка языка программирования Паскаль мы решили прислушаться к часто высказывавшемуся пожеланию перевести наши программы на язык C++. Если не рассматривать такие системозависимые языки, как Visual Basic, то C и C++ являются на сегодня самыми популярными и повсеместно используемыми языками. Они имеются практически на всех компьютерных платформах, а их популярность обеспечивается удобством при разработке приложений, тесно взаимодействующих с операционной системой. Кроме того, объектно-ориентированные возможности C++ делают его особенно подходящим для создания графических пользовательских интерфейсов.

Перерабатывая прежние программы, написанные на Паскале, мы не ограничивались простым переводом их на другой язык, но постарались создать полноценную новую реализацию с использованием множества возможностей, которые предоставляет язык C++. Мы надеемся, что именно таким способом сумеем предоставить читателю современный и мощный набор программ и библиотек, не жертвуя при этом понятностью кода, которая обеспечила успех прежним редакциям.

Изменения в содержании книги вызваны (наряду с другими причинами) последовательным введением векторно-матричных обозначений, которые позволяют компактно и современно описывать любые преобразования координат. Соответствующие классы и операции C++ позволяют удобно преобразовывать эти обозначения в программный код. При необходимости мы вносили изменения в соответствии с принятыми на сегодня соглашениями и уточненными значениями величин. В частности, это касается уточнения параметров вращения планет и вычисления географической долготы на Земле. Теперь в астрономии, как и в географии и спутниковой геодезии, долгота всегда считается положительной к востоку. С учетом этого был, например, изменен знак при вычислении звездного времени и коэффициенты станции при определении обстоятельств покрытия звезд Луной. Все эти изменения внесены в соответствующие алгоритмы и программы.

Глава, посвященная определению орбит по трем наблюдениям, была дополнена описанием гауссова метода определения орбит, благодаря чему случаи, допускающие несколько решений, теперь не составляют проблемы. Добавлена программа Phases, которая определяет даты возможных солнечных и лунных затмений, а также рассчитывает фазы Луны. Множество усовершенствований было сделано для того, чтобы программы стали более дружественными к пользователю. Например, при вычислении восходов и заходов теперь можно выбирать между гражданскими, навигационными и астрономическими сумерками, а при вычислении обстоятельств звездных покрытий используется набор критериев для того, чтобы пометить трудные для наблюдения события. Для всех программ при необходимости в командной строке можно задать входной и выходной файлы.

На прилагаемом CD-ROM впервые оказалось возможно разместить в неупакованном виде исполняемые программы для PC, работающие под Windows и Linux. Это даст возможность немедленно начать пользоваться программами и данными даже тем читателям, у кого нет компилятора C++ и опыта программирования. Компилятор понадобится только тем, кто захочет адаптировать программы под свои личные нужды. Для них в приложении приводятся советы по установке и реализации.

В качестве бонуса на диске размещены каталог положений и собственных движений звезд «Position and Proper Motion Catalogue» С. Рёзера (S. Rцser) и У. Бастиана (U. Bastian) (опубликованный Spectrum Akademie Verlag), который содержит информацию примерно о 470 000 звезд, а также база данных элементов орбит астероидов «Asteroid Orbital Elements», подготовленная Е. Боуэллом и содержащая информацию более чем о 50 000 малых планет. Соответствующие файлы предоставлены для всеобщего доступа Гейдельбергским институтом астрономических вычислений (Astronomische Rechen-Institute) и Лоувелловской обсерваторией, которым — а также лично авторам — мы выражаем нашу благодарность. Для упрощения использования этих очень крупных каталогов на диске размещен ряд дополнительных программ, позволяющих отбирать данные для ввода в программы Foto и Numint.

Мы также советуем читателям посетить web-сайт http://www.springer.de/phys/books/astropc/, где мы совместно с издателем размещаем полезную информацию, относящуюся к этой книге. В частности, вы найдете здесь исходные тексты программ на Паскале из предыдущей редакции «Астрономии на персональном компьютере», а также ссылки на полезные Интернет-ресурсы. При необходимости здесь также будут размещаться исправленные и обновленные версии программ.

Мы хотим выразить нашу признательность докторам Ч. Карону (Ch. Caron), К.-Д. Бачему (C.-D. Bachem) и Б. Райшель-Майеру (B. Reichel-Mayer) из Springer-Verlag в Гейдельберге за сотрудничество и поддержку в ходе подготовки этой книги. Мы также благодарим доктора Л. Вейдингера (L. Weidinger), чья помощь позволила перенести наши программы в среду Linux.

О. Монтенбрук и Т. Пфлегер
Январь 2000

Предисловие авторов ко второму изданию

После выхода первого издания «Астрономии на персональном компьютере» мы получили огромное количество откликов. Вместе с заинтересованностью издателя это дало нам импульс к подготовке новой значительно дополненной редакции.

Первое важное дополнение — это глава о вычислении возмущений. В ней описывается, как учесть возмущения со стороны планет при расчете движения астероидов и комет. Представленная в книге программа Numint позволяет вычислять их положения значительно точнее, чем раньше, что бывает очень важно при поиске предельно слабых объектов. Инструментарий по расчету возмущенного движения дополняет главы, посвященные вычислению эфемерид, определению орбит и астрометрии.

В другой новой главе обсуждается вычисление физических эфемерид планет и Солнца, что восполняет существенный пробел в первом издании. Теперь любители астрономии получают в свое распоряжение средства для предвычисления вида планет, а также для упрощения последующей обработки своих наблюдений.

Другие изменения касаются вычисления моментов восхода и захода планет и местных обстоятельств солнечных затмений.

И последнее, теперь существует лишь одна версия прилагаемой к книге дискеты. После того как фирма Applicaion Systems Heidelberg выпустила для Atari ST/TT свой компилятор Pure Pascal, совместимый с Turbo Pascal фирмы Borland, больше нет необходимости в специальной версии дискеты для компьютеров Atari. Прилагаемая дискета может, таким образом, без изменений использоваться как с компилятором Turbo Pascal на компьютерах, совместимых с IBM PC, так и с компилятором Pure Pascal на компьютерах Atari. Подробности, касающиеся процедуры установки, можно найти на дискете в файле aareadme.doc.

Мы выражаем благодарность издательству Springer-Verlag за помощь в нашей работе, а также Application Systems Heidelberg за оказанную техническую поддержку.

О. Монтенбрук и Т. Пфлегер
Мюнхен, июль 1994

Предисловие авторов к первому изданию

В наше время всякий, кто имеет дело с астрономическими вычислениями, независимо от того, профессионал он или любитель, непременно прибегает к помощи компьютера. Благодаря повсеместному распространению персональных компьютеров из этого утверждения практически нет исключений. Теперь, не выходя из-за стола, можно делать такие вычисления, которые еще недавно вообще нельзя было выполнить за разумное время. Но рост технических возможностей компьютеров привел к тому, что выросла и потребность в мощных, то есть быстрых и точных, программах для них. Вполне понятно желание везде, где это возможно, обходиться без использования традиционных астрономических ежегодников. Поэтому мы с большим воодушевлением восприняли предложение нашего издателя подготовить книгу об основных принципах сферической астрономии, эфемеридных вычислений и небесно-механических расчетов.

В «Астрономии на персональном компьютере» читатели, разрабатывающие свои собственные программы, найдут обширную библиотеку процедур на языке программирования Паскаль, которые позволяют решать многочисленные отдельные расчетные задачи. В число этих процедур входят часто применяемые преобразования координат, календарные вычисления, решение задачи двух тел. Отдельные процедуры позволяют вычислять точные положение Солнца, Луны, а также планет Солнечной системы с учетом их взаимных возмущений. Благодаря широкому распространению языка Паскаль, а также исключению операций, специфичных для конкретных вычислительных систем, приведенные программы можно использовать на самых разных компьютерах — от персоналок до крупнейших мэйнфреймов. Наличие готовых процедур должно освободить читателей от необходимости изобретать колесо и позволит им сконцентрироваться на своих собственных содержательных задачах.

Каждая глава этой книги посвящена строго ограниченной теме и заканчивается готовой самостоятельно работающей программой. Начиная с относительно простых вопросов — определения моментов восхода и захода, вычисления положений планет, мы переходим к более сложным темам, таким как предвычисление солнечных затмений и покрытий звезд. Программы для астрометрической редукции фотографий звездных полей и для определения орбит дают возможность пользователю самостоятельно определять элементы орбит комет и астероидов. Те же читатели, у кого нет программистского опыта, могут использовать соответствующие готовые приложения.

Для того чтобы читатели могли лучше понять работу программ, в книге рассматриваются необходимые вопросы из астрономии и математики, лежащие в ос-

нове решения конкретных задач. Знакомство с этими вопросами позволяет также адаптировать программы для собственных нужд. Эта тесная связь между теорией и практикой позволяет порой гораздо проще, чем в классических учебниках, объяснять некоторые сложные вопросы. Подводя итоги, мы надеемся, что читатели получат надежный базовый инструментарий для астрономических расчетов на своем компьютере.

Мы благодарим С. Данлопа за великолепный перевод и докторов Г. Волшина (G. Wolschin) и К.-Д. Бачема (C.-D. Bachem) из Springer-Verlag за активное сотрудничество и проявленную заинтересованность в ходе подготовки этой книги к публикации. Мы также выражаем благодарность всем нашим друзьям и коллегам, чьи идеи и советы, а также помощь в корректуре рукописи и тестировании программ были очень важны для успеха этой книги.

О. Монтенбрук и Т. Пфлегер
Мюнхен, июль 1990

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу электронной почты comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На web-сайте издательства http://www.piter.com вы найдете подробную информацию о наших книгах.

1. Введение

В последние годы мощность персональных компьютеров постоянно растет, а цена их снижается. Благодаря этому компьютеры стали доступны многим из тех, кто интересуется астрономией. Сама собой появляется мысль использовать их для астрономических вычислений. Однако какие преимущества может дать использование собственного компьютера, когда все наиболее важные для наблюдений данные, полные и точные, можно найти, например, в астрономическом календаре или ежегоднике?

1.1. Несколько примеров

Давайте для начала займемся определением моментов восхода и захода Солнца и Луны. Нельзя игнорировать тот факт, что при перемещении наблюдателя из одной точки Земли в другую эти моменты будут меняться. В то же время в астрономических календарях моменты восхода и захода обычно указаны только для одного конкретного места (обычно для широты Москвы). Для того чтобы определить эти моменты в других точках, обычно рекомендуется проводить интерполяцию или определять поправки по номограммам. Гораздо удобнее поручить работу по вычислению восходов и заходов компьютеру. Да и точность при этом будет выше.

В большинстве ежегодников значительная часть объема занята таблицами положений Солнца, Луны и планет. Из всей этой информации наблюдателю на практике оказывается нужна лишь очень малая часть. Пользуясь подходящей программой, можно получать только необходимые данные, причем представленные в нужной системе координат.

Еще важнее иметь возможность самостоятельно рассчитывать положения других небесных тел. Эфемериды вновь открытых объектов часто недоступны или появляются с задержкой. Однако достаточно, например, получить элементы орбиты кометы, чтобы с хорошей точностью рассчитать ее путь на небесной сфере.

Солнечное затмение — настолько захватывающее небесное явление, что для его наблюдения многие любители астрономии предпринимают далекие путешествия. И хотя официальные источники могут предоставлять весьма подробную информацию о конкретном затмении, часто поездку нужно планировать задолго до публикации этих данных. А если к тому же нужный вам населенный пункт не упоминается в астрономическом календаре, то потребность в специальной программе, вычисляющей местные обстоятельства затмения, становится особенно насущной.

Всякому, кто захочет наблюдать покрытия звезд Луной, потребуется знать моменты контактов для того места, откуда он ведет наблюдения. Хотя в календарях есть соответствующие данные, они обычно приводятся только для крупных городов. При наблюдениях из других мест на них можно полагаться лишь как на приближенные значения. Приводимая в этой книге программа Occult, позволяет как предсказывать покрытия звезд, так и с высокой точностью рассчитывать их обстоятельства для произвольно выбранной группы звезд.

Многие любители астрономии увлекаются астрофотографией. Пользуясь опорными звездами с известными координатами, можно самостоятельно определить по фотографии координаты астероида или кометы. Если удалось получить, по крайней мере, три фотографии с интервалом во времени, то по ним можно определить элементы орбиты объекта. Программы Photo, Gauss и Comet представляют собой набор инструментов для проведения этих весьма сложных вычислений. Наблюдателям астероидов должна понравиться программа Numint, которая вычисляет точные эфемериды с учетом гравитационных возмущений больших планет. Она также полезна, когда имеющиеся элементы орбиты были определены несколько лет назад.

1.2. Астрономия и вычисления

Любой разговор об астрономических вычислениях и программах будет несерьезным, без обсуждения точности получаемых результатов. Конечно, прежде всего, точность программ определяется математическим и физическим описанием задачи, а также качеством используемых алгоритмов. Часто ошибочно считают, что использование вычислений с двойной точностью автоматически гарантирует точность работы программы. Однако даже правильно написанная программа может давать неверные результаты, если использованные в ней вычислительные методы сами по себе неточны.

В качестве примера можно привести расчет эфемерид на основе решения задачи двух тел, когда орбита планеты упрощенно представляется эллипсом. Кроме гравитационного притяжения со стороны Солнца, которое определяет форму орбиты, есть также воздействия со стороны других планет, которые приводят к периодическим и вековым возмущениям. Для нижних планет эти возмущения могут достигать примерно одной угловой минуты, а для верхних — целого градуса. Теоретические кеплеровы орбиты — лишь приближение к реальному положению дел. Если нам требуется высокая точность, то совершенно необходимо использовать более сложную теорию движения планет, учитывающую их взаимные возмущения. Использование арифметики высокой точности даже в малой степени не поможет справиться с этой ситуацией.

При разработке программ мы стремились к достижению примерно такой же точности, какая характерна для астрономических ежегодников. Ошибки в фундаментальных процедурах определения координат Солнца, Луны и планет имеют величину порядка 1–3". Такая точность требуется для предвычисления солнечных затмений и покрытий звезд, а значит, и для большинства других задач.

Всякий, кто хочет быть последовательным при выполнении точных вычислений, должен предварительно принять четкие определения относительно используемых систем координат. Опыт показывает, что сравнение значений, отнесенных к разным системам координат, приводит к весьма забавным результатам. Порой приходится слышать возмущенные отклики, что, мол, программа дает неточные результаты, поскольку они отличаются от тех, что приведены в ежегоднике. Часто причина таких расхождений состоит в небольшом различии базисов используемых систем координат, которого пользователь не заметил и поэтому считает, что обнаружил ошибку.

Именно поэтому мы оговариваем все существенные моменты и поправки, такие как прецессия, нутация или аберрация света. Мы часто упоминаем об отличии всемирного времени от эфемеридного, из-за которого регулярно возникают трудности. Частое упоминание этих эффектов вызывать у вас раздражение. Но мы делаем это для того, чтобы помочь как можно лучше оценить величину и практическое значение каждой из поправок непосредственно перед использованием данной конкретной программы. Встречающиеся повторения мы считаем необходимыми, поскольку каждая глава книги — и даже отдельная специальная тема — может читаться независимо от других. Однако поскольку программы часто строятся с использованием одних и тех же процедур, читателю рекомендуется все-таки познакомиться с другими главами, имеющими отношение к рассматриваемому вопросу.

Достоинства программы зависят не только от математического описания задачи, но также и от использования подходящих вычислительных методов. Этот вопрос затрагивается время от времени в контексте обсуждения различных приложений. В качестве примера можно привести вычисление функции угла при расчете периодических возмущений орбит Луны и планет.

Благодаря использованию теоремы о суммировании и рекурсивных соотношений для тригонометрических функций удалось существенно сократить время, требуемое для работы программ Planpos и Luna. В тех случаях, когда для небесного тела по каким-либо причинам требуется получить большое количество положений, разделенных короткими интервалами времени, рекомендуется использовать аппроксимацию полиномами Чебышева. Они позволяют построить — для ограниченного отрезка времени — простое и быстро вычисляемое выражение, которое дает возможность без потери точности определять нужные величины. Этот прием применяется в программах Eclipse, EclTimer и Occult. Другие аспекты вычислительной математики обсуждаются в связи с реализацией метода Ньютона для решения уравнений, регуляризации ошибок и квадратичной интерполяцией. Наконец, в связи с задачей астрометрической редукции фотопластинок рассматривается простой, но устойчивый численный алгоритм аппроксимации по методу наименьших квадратов.

1.3. Приемы и языки программирования

Характерный для последний лет феноменально быстрый рост аппаратной мощности компьютеров привел к значительному прогрессу в сфере программного обеспечения. Прикладные программы, становясь все более мощными и удобны-

ми, в большинстве своем используют теперь графический интерфейс Microsoft Windows. Однако многое изменилось и с точки зрения программиста. Стало желательно и даже необходимо разрабатывать графические приложения, управляемые событиями, генерируемыми в ответ на действия пользователя. В разработке программного обеспечения нормой стали объектно-ориентированные методы. Новые тенденции также возникли под влиянием практики создания приложений для бизнеса. Программное обеспечение должно быть пригодно для работы в сети. Оно должно проектироваться и создаваться по возможности независимо от аппаратной платформы и конкретной операционной системы. Одним из следствий их требований стало широкое распространение объектно-ориентированных языков программирования, таких как C++ и, в последнее время, Java (который очень похож на C++). Преимущества последнего проявляются в основном при обработке данных в среде корпоративной интрасети или в Интернете.

В сфере разработки технических и научных приложений по-прежнему применяются структурные языки Fortran и Ada, которые, правда, в последних версиях Fortran 90 и Ada 95 подверглись значительной модернизации. И все-таки, даже в этой сфере объектно-ориентированный язык C++ завоевывает новых приверженцев. Это в основном происходит благодаря его универсальности, отличной производительности скомпилированных программ и повсеместной доступности. C++ поддерживает как структурное, так и объектно-ориентированное программирование и поэтому его обычно характеризуют как гибридный язык. Конечно, астрономические и численные алгоритмы лишь в очень ограниченной мере используют преимущества объектно-ориентированного программирования. Однако если эти алгоритмы применяются в программах с графическим пользовательским интерфейсом, то C++ в полной мере проявляет себя как гибкий и мощный язык, позволяющий с равной эффективностью реализовывать и алгоритмы, и пользовательский интерфейс. Вот почему для нового издания этой книги мы остановили свой выбор на языке программирования C++.

Здесь надо сказать, что по сравнению с другими языками C++ имеет много недостатков, которые затрудняют разработку сложных научных программ. В частности, у него слабая поддержка одномерных и многомерных массивов, нет локальных процедур, неудачная концепция модуля, не поддерживается проверка индексов массивов, а также выполняется неконтролируемое преобразование типов.

С учетом всех этих проблем, а также предоставляемых языком возможностей, мы использовали при разработке программ элементы обоих подходов — структурного и объектно-ориентированного. Как и в прежней редакции, где использовался язык Pascal, самой главной целью было создание прозрачных и понятных реализаций базовых алгоритмов, а также обеспечение модульности и возможности повторно использовать готовые компоненты.

При разработке астрономических программ мы постоянно сталкиваемся с теми или иными вспомогательными базовыми задачами. В качестве примеров можно привести вычисление математических функций, преобразование координат между различными системами, определение точных положений планет, Солнца или Луны. Было бы крайне нерационально каждый раз заново решать

эти задачи. Если нужные функции включены в состав легко доступной библиотеки готовых и проверенных функций, тогда при разработке можно сконцентрироваться на специфических особенностях создаваемой программы и тем самым гораздо быстрее и проще достигнуть своей цели.

Многие операции, функции и классы, которые представлены в данной книге, составляют основу мощной библиотеки астрономических программ. В ее состав входят следующие отдельные модули:

- процедуры ввода-вывода (APC_IO.cpp);
- математические и астрономические константы (APC_Const.cpp);
- научные и технические функции и классы (APC_Math.cpp, APC_VecMat3D.cpp);
- классы и функции для решения отдельных математических задач (APC_Cheb.cpp, APC_DE.cpp);
- функции и специальные типы для работы со временем и календарем (APC_Time.cpp);
- различные функции из сферической астрономии (APC_Spheric.cpp);
- специальные функции для расчета прецессии и нутации (APC_PrecNut.cpp);
- функции для вычисления эллиптических, параболических и гиперболических кеплеровских орбит (APC_Kepler.cpp);
- функции для вычисления точных положений Солнца, Луны и планет с учетом различных возмущений (APC_Sun.cpp, APC_Moon.cpp, APC_Planets.cpp);
- функции для вычисления физических эфемерид планет (APC_Phys.cpp).

Отдельные компоненты этой библиотеки создаются и объясняются последовательно по мере необходимости в соответствующих главах книги. Нужные модули можно легко подключить к прикладной программе, используя соответствующие заголовочные файлы (APC_*.h).

В использовании классов и объектов мы ограничились отдельными специальными случаями, поскольку злоупотребление этими элементами языка может сделать программы более трудными для чтения. В частности, огромную пользу принесло определение для векторов и матриц классов Vec3D и Mat3D, использующих перегрузку операций. C++ позволяет перегружать такие операции, как +, – или *, так что они начинают работать с определенными пользователем типами данных. Благодаря этой возможности сложение векторов или умножение матриц записываются гораздо короче и понятнее, чем с использованием вызовов функций. Использование этих классов позволяет достичь близкого подобия между инструкциями программ и математическими уравнениями, что делает код значительно более ясным и удобочитаемым.

Кроме того, мы посчитали важным определить вспомогательные классы Time, DataTime и Angle для упрощения форматированного вывода значений времени и углов. Эти классы поддерживают использование операции сдвига (<<), которая традиционно применяется в C++ для записи операции вывода.

Примером полезного применения объектно-ориентированных методов в численных процедурах могут служить классы SolverDE, предназначенные для числен-

ного интегрирования дифференциальных уравнений, и SolverLSQ — для построения линейной регрессии. Хотя отдельные шаги вычислений не отличаются от реализации на классических языках программирования, концепция классов обеспечивает высокий уровень абстракции и оптимальную инкапсуляцию данных. Пользователю больше не нужно заботиться о подготовке необходимых рабочих массивов, поскольку эту работу выполняет конструктор соответствующего объекта.

2. Системы координат

Для задания положений звезд и планет в астрономии используется целый ряд различных систем координат. Прежде всего, надо различать гелиоцентрические координаты, которые задаются относительно Солнца и геоцентрические координаты, связанные с Землей. Эклиптические координаты задают положение точки относительно плоскости земной орбиты, тогда как экваториальные отсчитываются от земного или небесного экватора. Вследствие прецессии плоскость экватора медленно поворачивается и поэтому, задавая экваториальные координаты, нужно указывать, к какой эпохе они относятся. Казалось бы, можно раз и навсегда выбрать для всех расчетов единую общую систему координат и всегда пользоваться ею. Однако на практике каждая из координатных систем имеет свои особые преимущества, которые делают ее более подходящей для определенных задач.

Поскольку существуют различные системы координат, часто возникает необходимость переводить координаты из одной системы в другую. Для этой цели предназначена программа Coco (Coordinate Conversion — преобразование координат). Она выполняет точное преобразование между эклиптическими и экваториальными, а также между геоцентрическими и гелиоцентрическими координатами и при этом учитывает различие эпох, к которым относятся координаты. Для каждого преобразования в программе предусмотрена отдельная функция, сопровождаемая коротким описанием. Эти функции лежат в основе многих других программ, и мы постоянно будем пользоваться ими в дальнейшем.

2.1. Приступаем

Хотя в состав стандартной библиотеки языка C++ входит множество математических функций, для астрономических расчетов их все-таки недостаточно. Поэтому мы начнем с составления собственной библиотеки инструментальных модулей. Первый из таких модулей APC_Const содержит определения ряда важных констант, которые неоднократно будут использоваться в дальнейшем. Модуль состоит из одного только файла-заголовка APC_Const.h, который может подключаться к другим модулям при помощи директивы #include. В него включены математические постоянные, например число π и коэффициенты для перевода угловых величин из одних единиц измерения в другие, а также астрономические и физические константы, например значения астрономической единицы и скорости света.

```
const double pi        = 3 14159265358979324;
const double pi2       = 2.0*pi;
const double Rad       = pi / 180.0;
```

```
const double Deg       = 180.0 / pi;
const double Arcs      = 3600.0*180.0/pi;
...
const double AU        = 149597870.0;    // Астрономическая единица [км]
const double c_light   = 173.14;         // скорость света [а.е./день]
```

В модуле APC_Math наряду с другими имеется две функции

```
//------------------------------------------------------------------------------
// Frac: Возвращает дробную часть числа
//------------------------------------------------------------------------------
double Frac (double x)
{
  return x-floor(x);
}

//------------------------------------------------------------------------------
// Modulo: Вычисляет остаток от деления x на y
//------------------------------------------------------------------------------
double Modulo (double x, double y)
{
  return y*Frac(x/y);
}
```

которые позволяют вычислять дробную и целую части числа. Эти функции часто применяются, например, когда нужно приводить монотонно растущую долготу небесного тела к диапазону от 0 до 360°.

Две другие полезные функции из модуля APC_Math — Ddd и DMS – позволяют работать с традиционным (шестидесятеричным) представлением времени и углов. При задании углов в градусах доли градуса принято выражать в минутах и секундах дуги, а при указании времени используются обычные минуты и секунды. Однако пользуются таким представлением обычно только при вводе и выводе данных, так как по ходу вычислений значение времени или величину угла хранят в виде одного вещественного числа. Преобразование из внутреннего формата в градусы минуты и секунды не вызывает никаких трудностей для положительных значений. Однако при выводе отрицательных величин важно корректно обрабатывать знак. Работу этих двух функций лучше всего объяснить на примерах:

```
   Dd          DMS
15.50000   15 30 00.0
-8.15278   -8 09 10.0
 0.01667    0  1  0.0
-0.08334    0 -5  0.0
```

При преобразовании отрицательных чисел Dd в градусы, минуты и секунды только старшее (ненулевое) значение из трех чисел D, M и S будет отрицательным. Также надо заметить, что в обеих функциях значения D и M, которые по самой своей природе являются целыми, заданы переменными типа int, тогда как S является переменной типа double.

```
//------------------------------------------------------------------------------
// Ddd: Преобразование углов из градусов, минут и секунд дуги
//      в десятичное представление величины угла
// D,M,S    Градусы, минуты и секунды дуги
// <return>:  Угол в десятичном представлении
```

```
//------------------------------------------------------------------------
double Ddd(int D. int M. double S)
{
  double sign;
  if ( (D<<0) || (M<<0) || (S<<0) ) sign = -1.0; else sign = 1.0.
  return  sign * ( fabs(D)+fabs(M)/60.0+fabs(S)/3600.0 );
}
//------------------------------------------------------------------------
// DMS: Вычисляет градусы. минуты и секунды дуги по заданному значению угла
//   Dd       Угол в градусах в десятичном представлении
//   D.M.S    Градусы. минуты и секунды дуги
//------------------------------------------------------------------------
void DMS (double Dd. int& D. int& M. double& S)
{
  double x;
  x = fabs(Dd);   D = int(x);
  x = (x-D)*60.0; M = int(x);  S = (x-M)*60 0;
  if (Dd<<0.0) { if (D!=0) D*=-1; else if (M!=0) M*=-1. else S*=-1.0; }
}
```

Поскольку преобразование углов в градусы, минуты и секунды требуется почти исключительно при осуществлении вывода данных, эти операции удобно объединить. Воспользуемся возможностями языка C++ и определим соответствующий класс:

```
// Тег формата для вывода Angle (используется в операции вывода класса Angle)
enum AngleFormat {
  Dd.    // десятичное представление
  DMM.   // градусы и целые минуты дуги
  DMMm.  // градусы и минуты дуги в десятичном представлении
  DMMSS. // градусы. минуты и целые секунды дуги
  DMMSSs // градусы и минуты дуги с секундами в десятичном представлении
}:

//
// Вспомогательный класс для вывода углов в шестидесятеричном формате
//
class Angle
{
  public·
    // Конструктор
    Angle (double alpha. AngleFormat Format=Dd).
    // Модификаторы
    void Set (AngleFormat Format=Dd).
    // Вывод угла
    friend std.·ostream& operator <<<< (std::ostream& os. const Angle& alpha);
  private:
    double       m_angle;
    AngleFormat  m_Format;
}.
```

Объекты класса `Angle` создаются вызовом одноименного конструктора с указанием значения угла и дескриптора формата. Эти данные сохраняются в полях объекта и в дальнейшем используются при выполнении операции вывода <<. Выводимое значение угла сначала преобразуется в градусы, минуты и секунды в соответствии с дескриптором формата, а затем каждое из полученных значений вы-

водится с учетом заданного форматирования. Реализация операции вывода << совместима с использованием стандартных манипуляторов для потокового вывода, которые позволяют задавать число позиций, режим вывода знака и другие форматные спецификации. Это позволяет легко подстраивать формат вывода под различные требования. Мы не станем больше задерживаться на реализации этого класса, так как она достаточно очевидна. В употреблении он очень прост, что наглядно демонстрирует следующая программа, выводящая значение угла 12 °3456 в пяти различных форматах:

```
#include <iostream>
#include <iomanip>
#include "APC_Math.h"
void main()
{
  using namespace std;
  double x=12 3456;
  cout << setprecision(2) << setw(12) << Angle(x,Dd) << endl,
  cout << setprecision(2) << setw(12) << Angle(x,DMM) << endl;
  cout << setprecision(2) << setw(12) << Angle(x DMMm) << endl,
  cout << setprecision(2) << setw(12) << Angle(x,DMMSS) << endl;
  cout << setprecision(2) << setw(12) << Angle(x,DMMSSs) << endl;
}
```

Результат работы этой программы выглядит так:

```
    12.35
    12 21
 12 20.74
    12 20 44
12 20 44 16
```

Знак (если он выводится) всегда будет выравниваться влево.

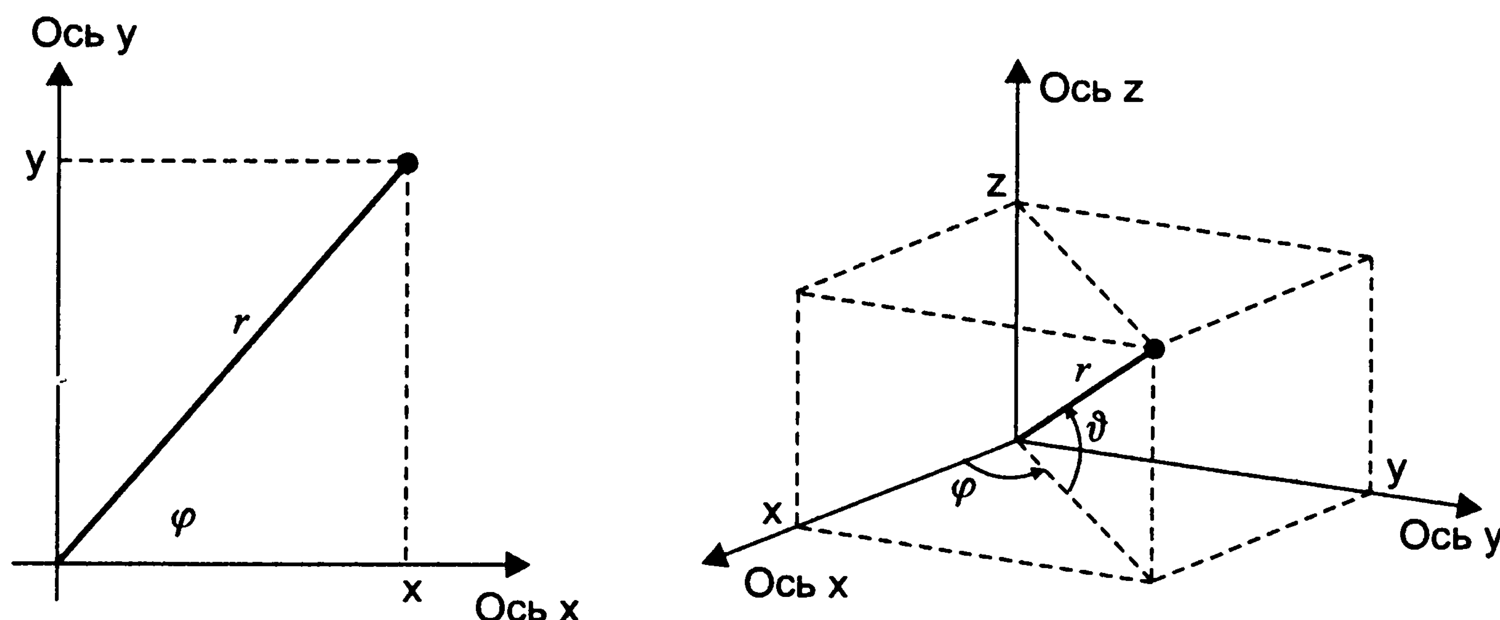


Рис. 2.1. Полярные координаты на плоскости и в пространстве

Наконец, перейдем к третьему модулю — библиотеке APC_VecMat3D, которая реализует классы трехмерных векторов и матриц, а также целый ряд связанных с ними операций и функций. Трехмерные векторы служат для задания положения точек $\boldsymbol{r}$ в пространстве. Причем наряду с декартовыми координатами $\boldsymbol{r} = (x, y, z)$ можно использовать полярные: r, ϑ, φ (рис. 2.1). Эти два вида координат связаны друг с другом следующими соотношениями:

$$
\begin{aligned}
x &= r\cos\vartheta\,\cos\varphi, & r &= \sqrt{x^2+y^2+z^2},\\
y &= r\cos\vartheta\,\sin\varphi, & \operatorname{tg}\varphi &= y/x,\\
z &= r\sin\vartheta, & \operatorname{tg}\vartheta &= z/\sqrt{x^2\;y^2}\,.
\end{aligned}
\tag{2.1}
$$

Класс Vec3D поддерживает оба представления, позволяя в любой момент совершенно свободно пользоваться тем из них, которое окажется более удобным. Для этого предусмотрены соответствующие конструкторы и операторы доступа []:

```
//
// Vec3D: Трехмерные векторы
//
class Vec3D
{
  public:
    // Конструкторы
    Vec3D (); // конструктор по умолчанию инициализирует вектор нулями
    Vec3D (double X, double Y, double Z);
    Vec3D (const Polar& polar);
    // доступ к компонентам (только чтение)
    double operator [] (index Index) const;
    // получение полярных углов или модуля вектора
    double operator [] (pol_index Index);
    // простой вывод вектора
    friend std::ostream& operator << (std::ostream& os, const Vec3D& Vec);
    ..
private:
    // Члены
    double m_Vec[3];        // компоненты вектора
    double m_phi;           // полярный угол (азимут)
    double m_theta;         // полярный угол (высота)
    double m_r;             // модуль вектора
    bool   m_bPolarValid;   // флаг корректности полярных координат
    // Вычисление полярных компонентов по требованию
    void CalcPolarAngles ();
};
```

Специальные индексные типы данных index, pol_index и Polar определены следующим образом:

```
enum index     { x=0, y=1, z=2 };
enum pol_index { phi=0, theta=1, r=2 }; // азимут, высота, радиус
struct Polar {
  // Конструкторы
  Polar();
  Polar(double Az, double Elev, double R = 1.0);
  // Members
  double phi;    // азимут вектора
  double theta;  // высота вектора
  double r;      // длина вектора
};
```

Преобразование декартовых координат в полярные выполняется приватной функцией CalcPolarAngles, которая автоматически вызывается при первом же обращении к полярным координатам с использованием операции []:

```
//
// Вычисление полярных координат
//
void Vec3D::CalcPolarAngles ()
{
  // Длина проекции на плоскость XY:
  const double rhoSqr = m_Vec[0] * m_Vec[0] + m_Vec[1] * m_Vec[1];
  // Модуль вектора
  m_r = sqrt ( rhoSqr + m_Vec[2] * m_Vec[2] );
  // Азимут вектора
  if ( (m_Vec[0]==0.0) && (m_Vec[1]==0.0) )
    m_phi = 0.0;
  else
    m_phi = atan2 (m_Vec[1], m_Vec[0]);
  if ( m_phi < 0.0 )  m_phi += 2.0*pi;
  // Высота вектора
  const double rho = sqrt ( rhoSqr );
  if ( (m_Vec[2]==0.0) && (rho==0.0) )
    m_theta = 0.0;
  else
    m_theta = atan2(m_Vec[2], rho);
}
```

После вызова этой функции переменной m_bPolar_Valid присваивается «истина», указывающее, что полярные координаты уже вычислены и при очередном обращении к ним функцию CalcPolarAngles вызывать не требуется. Надо заметить, что при вычислениях значения углов должны выражаться в радианах. Градусная мера углов используется только при вводе и выводе. В модуле APC_Const для пересчета радианов в градусы и обратно есть две константы: Deg = 180°/π и Rad = π/180°.

Наряду с основными конструкторами и операциями доступа к координатам класс Vec3D содержит множество операций и функций для быстрого и удобного проведения векторных вычислений. Например, векторы можно складывать, пользуясь операцией +, операция * позволяет умножать вектор на скаляр и т. п. (табл. 2.1).

Класс Vec3D дополняется классом трехмерных матриц Mat3D, который определен следующим образом:

```
//
// Mat3D: Матрицы преобразований координат в трехмерном пространстве
//
class Mat3D
{
  public:
    // Конструкторы (нулевая матрица; матрица из векторов-столбцов)
    Mat3D ();
    Mat3D ( const Vec3D& e_1, const Vec3D& e_2, const Vec3D& e_3 );
    // доступ к компонентам
    friend Vec3D Col(const Mat3D& Mat, index Index);
    friend Vec3D Row(const Mat3D& Mat, index Index);
    // простой вывод матрицы
    friend std::ostream& operator << (std::ostream& os, const Mat3D& Mat);
    // прочие операции и функции
    ...
```

```
  private:
    double m_Mat[3][3];  // Элементы матрицы
};
```

Таблица 2.1. Векторные и матричные операции модуля APC_VecMat3D

Имя	Arg_1 (a)	Arg_2 (a)	Значение (c)	Обозначение	Пояснение
-	Vec3D		Vec3D	$\boldsymbol{c}=-\boldsymbol{a}$	Унарный минус
	Mat3D		Mat3D	$\boldsymbol{C}=-\boldsymbol{A}$	
+	Vec3D	Vec3D	Vec3D	$\boldsymbol{c}=\boldsymbol{a}+\boldsymbol{b}$	Сложение векторов
	Mat3D	Mat3D	Mat3D	$\boldsymbol{C}=\boldsymbol{A}+\boldsymbol{B}$	Сложение матриц
*	double	Vec3D	Vec3D	$\boldsymbol{c}=a\boldsymbol{b}$	Умножение на скаляр
	Vec3D	double	Vec3D	$\boldsymbol{c}=\boldsymbol{a}b$	
	double	Mat3D	Mat3D	$\boldsymbol{C}=a\boldsymbol{B}$	
	Mat3D	double	Mat3D	$\boldsymbol{C}=\boldsymbol{A}b$	
/	Vec3D	double	Vec3D	$\boldsymbol{c}=\boldsymbol{a}/b$	Деление на скаляр
	Mat3D	double	Mat3D	$\boldsymbol{C}=\boldsymbol{A}/b$	
*	Mat3D	Vec3D	Vec3D	$\boldsymbol{c}=\boldsymbol{A}\boldsymbol{b}$	Матричное/векторное
	Vec3D	Mat3D	Vec3D	$\boldsymbol{c}=\boldsymbol{a}\boldsymbol{B}$	умножение
Norm	Vec3D		double	$c=\lvert\boldsymbol{a}\rvert$	Модуль вектора
Dot	Vec3D	Vec3D	double	$c=\boldsymbol{a}^T\boldsymbol{b}$	Скалярное произведение
Cross	Vec3D	Vec3D	Vec3D	$\boldsymbol{c}=\boldsymbol{a}\times\boldsymbol{b}$	Векторное произведение
Id3D			Mat3D	$\mathbf{1}$	Единичная матрица
R_x	double		Mat3D	$\boldsymbol{R}_x(a)$	Матрицы поворотов вокруг
R_y	double		Mat3D	$\boldsymbol{R}_y(a)$	осей базиса
R_z	double		Mat3D	$\boldsymbol{R}_z(a)$	

Объекты-векторы обычно задают положение в пространстве. Соответственно объекты-матрицы обычно служат для описания поворотов систем координат. Элементарные повороты вокруг осей x, y и z можно получить, пользуясь функциями R_x, R_y и R_z, которые принимают в качестве параметра значение угла поворота φ и вычисляют следующие матрицы поворотов:

$$\boldsymbol{R}_x(\varphi)=\begin{pmatrix} 1 & 0 & 0 \\ 0 & +\cos\varphi & +\sin\varphi \\ 0 & -\sin\varphi & +\cos\varphi \end{pmatrix}, \tag{2.2}$$

$$\boldsymbol{R}_y(\varphi)=\begin{pmatrix} +\cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ +\sin\varphi & 0 & +\cos\varphi \end{pmatrix}, \tag{2.3}$$

$$\boldsymbol{R}_z(\varphi)=\begin{pmatrix} +\cos\varphi & +\sin\varphi & 0 \\ -\sin\varphi & +\cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.4}$$

В этих формулах знак угла φ выбран так, чтобы положительные значения соответствовали положительному направлению поворота системы координат относительно оси вращения, то есть повороту против часовой стрелки.

2.2. Календарь и юлианские дни

При вычислении эфемерид часто требуется знать интервал времени между двумя определенными датами. Для этого удобно пользоваться последовательной нумерацией суток на протяжении очень длительных интервалов времени. В астрономии принято использовать дни юлианского периода, счет которых ведется от произвольно выбранного момента — 1 января 4713 года до н. э. В хронологических целях юлианские дни впервые использовал Джозеф Юстус Скалигер, а их название происходит от имени его отца Юлиуса Скалигера и ничего общего с юлианским календарем не имеет. На сегодняшний день юлианский период охватывает почти семь тысяч лет, и поэтому номера юлианских дней выражаются очень большими числами. Так, например, к полудню 23 июля 1980 года от начала периода прошло 2444444,0 дней. Если вспомнить, что секунда это примерно 0,00001 суток, то обнаруживается, что для точного задания времени в юлианских днях требуется 12 значащих цифр. При этом две старшие цифры меняются не чаще, чем раз в триста лет. Поэтому на практике часто пользуются модифицированными юлианскими датами:

$$MJD = JD - 2400000{,}5.$$

Модифицированные юлианские дни отсчитываются от полуночи 17 ноября 1858 года. В отличие от юлианских дней, которые начинаются в полдень, начало суток при подсчете MJD передвинуто на полночь и тем самым совпадает с началом обычных гражданских суток.

```
//------------------------------------------------------------------------------
// Mjd· Вычисление модифицированной юлианской даты по календарной дате и времени
//   Year. Month. Day  Компоненты календарной даты
//   Hour. Min. Sec    Компоненты времени (не обязательно)
//   <return>          Модифицированная юлианская дата
//------------------------------------------------------------------------------
double Mjd ( int Year. int Month. int Day.
             int Hour. int Min. double Sec )
{
  // Variables
  long    MjdMidnight;
  double  FracOfDay.
  int     b;


  if (Month<=2) { Month+=12. --Year.}
  if ( (10000L*Year+100L*Month+Day) <= 15821004L )
    b = -2 + ((Year+4716)/4) - 1179;     // Юлианский календарь
  else
    b = (Year/400)-(Year/100)+(Year/4).  // Григорианский календарь
  MjdMidnight = 365L*Year - 679004L + b + int(30 6001*(Month+1)) + Day.
```

```
  FracOfDay   = Ddd(Hour,Min,Sec) / 24.0;
  return MjdMidnight + FracOfDay;
}
```

Функция Mjd учитывает григорианскую реформу календаря, в соответствии с которой после 4 октября 1582 года (JD 2299159,5) следовала дата 15 октября 1582 года (JD 2299160,5)[1]. До этой реформы использовался юлианский календарь, в котором дата 29 февраля появлялась строго каждые четыре года. После реформы эту дату стали пропускать три раза в 400 лет — последние годы столетий, номера которых не делятся на 400, не считаются високосными. В результате средняя длина года по григорианскому календарю составляет

$$365 + 1/4 - 1/100 + 1/400 = 365{,}2425 \text{ суток,}$$

что немного меньше 365,25 суток, составляющих юлианский год.

Функция CalDat предназначена для определения обычной календарной даты по заданному номеру модифицированного юлианского дня. Приведем ее, поскольку она тесно связана с содержанием данного раздела.

```
//------------------------------------------------------------------------
// CalDat: Получение календарной даты и времени по модифицированной юлианской дате
//   Mjd             Модифицированная юлианская дата
//   Year,Month,Day  Компоненты календарной даты
//   Hour            Часы
//------------------------------------------------------------------------
void CalDat ( double Mjd,
              int& Year, int& Month, int& Day, double & Hour )
{
long    a,b,c,d,e,f;
  double  FracOfDay;
  // Преобразуем номер юлианского дня в календарную дату
  a = long(Mjd+2400001.0);
  if ( a < 2299161 ) {  // Юлианский календарь
    b = 0;
    c = a + 1524;
  }
  else {                // Григорианский календарь
    b = long((a-1867216 25)/36524.25);
    c = a +  b - (b/4) + 1525;
  }
  d     = long ( (c-122.1)/365.25 );
  e     = 365*d + d/4;
  f     = long ( (c-e)/30.6001 );
  Day   = c - e - int(30.6001*f);
  Month = f - 1 - 12*(f/14);
  Year  = d - 4715 - ((7+Month)/10);
```

[1] В этом вопросе нужно проявлять определенную осторожность, так как только Италия, Испания и Португалия приняли новый календарь в этот день. Большинство других католических стран континентальной Европы провели реформу в различные дни 1583 и 1584 годов, тогда как протестантские страны и города последовали этому примеру в основном в 1700–1701 годах. Великобритания и ее колонии провели реформу 3 сентября 1752 года (следующей датой было 14 сентября). Полный список приводится на с. 412–416 в книге *Explanatory Supplement to the Astronomical Ephemeris and the American Ephemeris and Nautical Almanac*, опубликованной совместно *U. S. Government Printing Office*, Washington и *Her Majesty's Stationery Office*, London, в 1974 году.

```
  FracOfDay = Mjd - floor(Mjd):
  Hour = 24.0*FracOfDay:
}
//---------------------------------------------------------------------------
// CalDat: Получение календарной даты и времени по модифицированной юлианской дате
//   Year.Month.Day  Компоненты календарной даты
//   Hour.Min.Sec    Компоненты времени
//---------------------------------------------------------------------------
void CalDat ( double Mjd. int& Year. int& Month. int& Day.
              int& Hour. int& Min. double& Sec )
{
  double Hours:
  CalDat (Mjd. Year. Month. Day. Hours).
  DMS (Hours. Hour. Min. Sec).
}
```

Два варианта функции CalDat (здесь используется возможность перегрузки имен функций в C++) отличаются лишь тем, в какой форме они возвращают информацию о дробной части суток. Первая выражает в часах в виде вещественного числа, а вторая представляет в часах, минутах и секундах. Последняя форма, как и представление углов в градусах, минутах и секундах, используется только при выводе значений времени. Продолжая аналогию с углами, модуль ATC_Time содержит два специальных класса Time и DateTime, предназначенных для упрощения вывода значений времени. Их конструкторы имеют следующие спецификации:

```
Time (double Hour. TimeFormat Format=HHMMSS):
DateTime (double Mjd. TimeFormat Format=None):
```

Тип TimeFormat

```
enum TimeFormat {
  None.   // не выводить время (только дату)
  DDd.    // выводить время в дробной части суток
  HHh.    // выводить время в форме часов и десятых долей часа
  HHMM.   // выводить время в форме часов и минут (округляя до следующей минуты)
  HHMMSS  // выводить время в форме часов. минут и секунд (округляя до следующей секунды)
}:
```

служит для задания формата и требуемого округления при выводе значений объектов Time и DateTime при помощи операции <<:

```
double MJD = Mjd ( 1961.01.14. 03.30.10.0 ): // 14 января 1961. 3.30:10
double JD = MJD + 2400000.5.                 // Юлианская дата
cout << setprecision(1)
     << "Date·" << DateTime(MJD.HHMMSS)      // Вывод
     << setprecision(3)
     << "MJD· " << setw(12) << MJD
     << "JD   " << setw(12) << JD.
```

2.3. Эклиптические и экваториальные координаты

Эклиптические и экваториальные координаты различаются основной координатной плоскостью, от которой производятся все отсчеты. В первом случае за плоскость (x,y) принимается плоскость земной орбиты, а во втором — плоскость,

перпендикулярная земной оси и параллельная земному экватору (рис. 2.2). Ось $x - x'$, общая для обеих систем, задает направление на точку весеннего равноденствия, обозначаемую ♈. Это направление перпендикулярно направлениям на северный полюс эклиптики (ось z) и на северный полюс мира (ось z'). Угол ε между плоскостями эклиптики и экватора составляет примерно 23°,5.

Как связаны эклиптические координаты точки (x, y, z) с экваториальными (x', y', z')? Поскольку $x = x'$, рассматривать нужно только преобразование $(y, z) \leftrightarrow$ $\leftrightarrow (y', z')$. Точка на оси y ($z = 0$) имеет экваториальные координаты $y' = +y \cos \varepsilon$ и $z' = +y \sin \varepsilon$. Если точка лежит на оси z, то ее экваториальные координаты будут $y' = -z \sin \varepsilon$ и $z' = +z \cos \varepsilon$. Таким образом, произвольная точка (x, y, z) в экваториальной системе будет иметь координаты

$$\begin{aligned} x' &= +x, \\ y' &= +y \cos \varepsilon - z \sin \varepsilon, \\ z' &= +y \sin \varepsilon + z \cos \varepsilon. \end{aligned} \tag{2.5}$$

Соответствующие обратные соотношения имеют вид

$$\begin{aligned} x &= +x', \\ y &= +y' \cos \varepsilon + z' \sin \varepsilon, \\ z &= -y' \sin \varepsilon + z' \cos \varepsilon. \end{aligned} \tag{2.6}$$

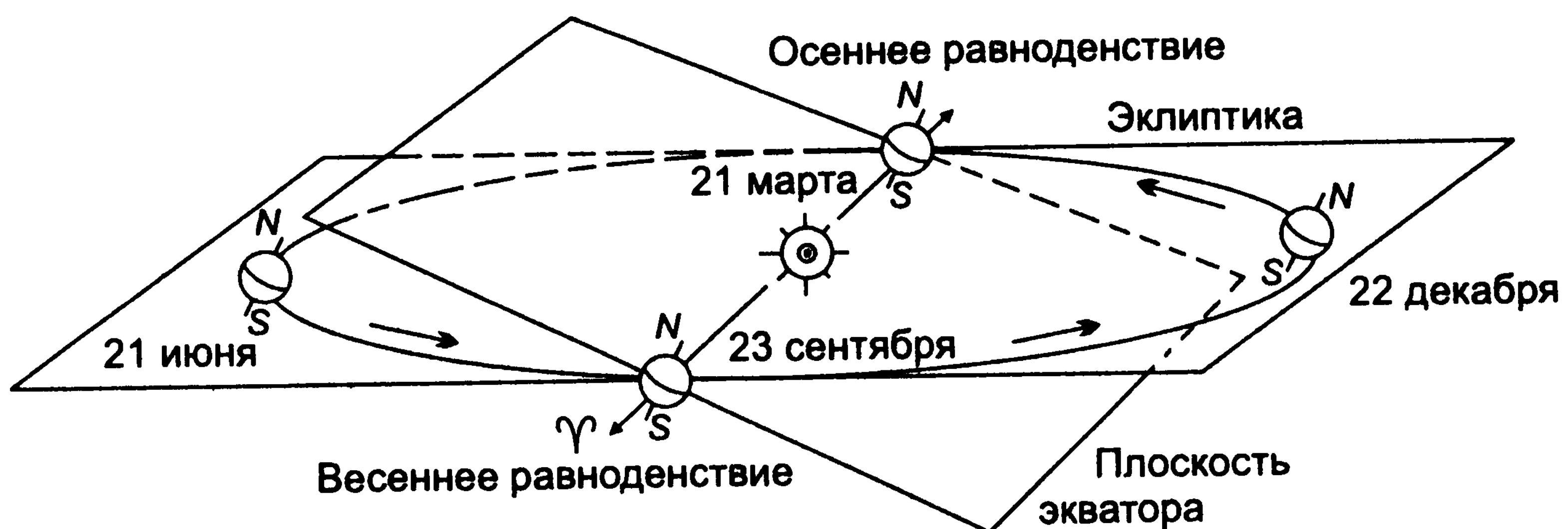


Рис. 2.2. Эклиптика и экватор

С прямоугольными координатами (x, y, z) связаны полярные координаты: эклиптическая долгота l, эклиптическая широта b и расстояние r (рис. 2.3).

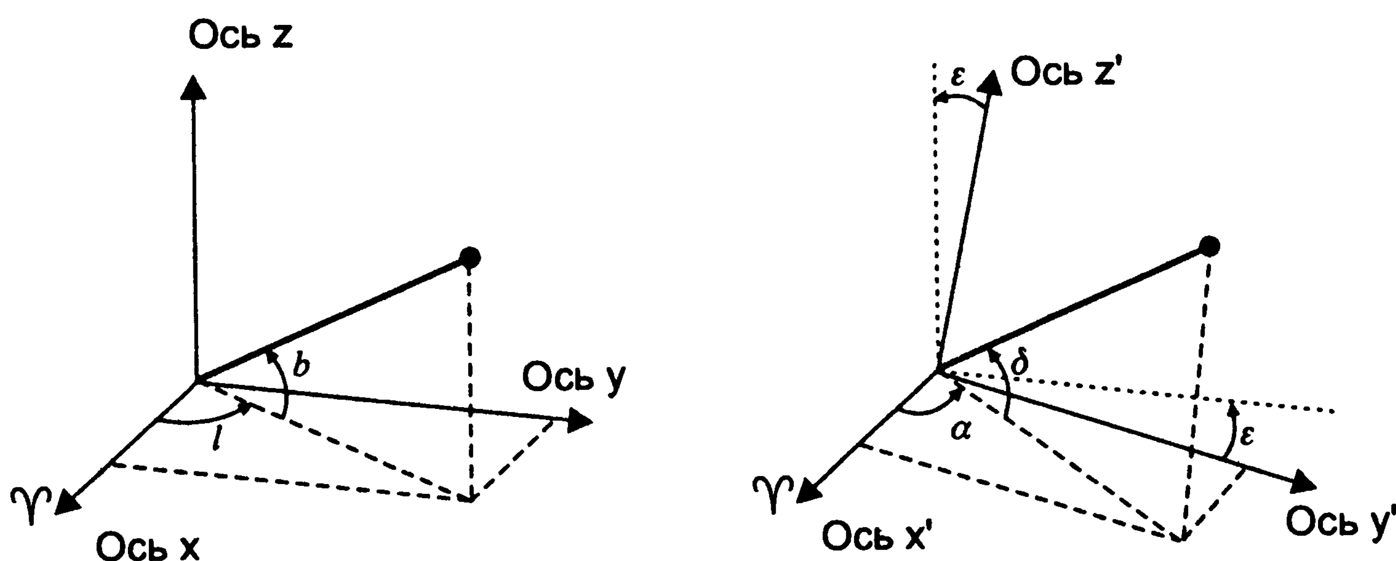


Рис. 2.3. Эклиптические и экваториальные координаты

$$
\begin{aligned}
x &= r\cos b\cos l, & x' &= r\cos\delta\cos\alpha,\\
y &= r\cos b\sin l, & y' &= r\cos\delta\sin\alpha,\\
z &= r\cos b, & z' &= r\sin\delta.
\end{aligned}
\tag{2.7}
$$

Соответствующие экваториальные координаты представлены прямым восхождением α, склонением δ и вновь расстоянием r, которое, естественно, совпадает в обеих системах координат.

Преобразование между эклиптическими и экваториальными координатами особенно ясно и компактно выглядит в векторно-матричной записи. Пользуясь матрицами элементарных поворотов, которые мы определили раньше, получаем

$$\boldsymbol{r} = \boldsymbol{R}_x(+\varepsilon)\boldsymbol{r}' \quad \text{и} \quad \boldsymbol{r}' = \boldsymbol{R}_x^T(+\varepsilon)\boldsymbol{r} = \boldsymbol{R}_x(-\varepsilon)\boldsymbol{r}, \tag{2.8}$$

где $\boldsymbol{R}_x^T$ обозначает транспонированную матрицу $\boldsymbol{R}_x$. Для выполнения этих преобразований в программе вектор, представленный объектом типа Vec3D, умножается на соответствующую матрицу поворота. Приведем небольшой фрагмент программы:

```
double eps = Rad*23.5;                  // Наклон эклиптики (в радианах)
Vec3D  e   = Vec3D(1.0,2.0,3.0);        // Эклиптические координаты
Vec3D  a   = R_x(eps)*e;                // Экваториальные координаты
cout << "l   " << Deg*e[phi]      << " b   " << Deg*e[theta] << endl
     << "RA " << Deg*a[phi]/15.0 << " Dec " << Deg*a[theta] << endl;
```

В этом примере значение прямого восхождения выводится, как это обычно принято, в часах ($1^h \equiv 15°$).

До сих пор мы не привели точного значения наклона эклиптики ε. Эта величина не является постоянной, каждое столетие она уменьшается примерно на 47″. В основном это происходит из-за медленного изменения земной орбиты под действием возмущений со стороны других планет. При задании координат их всегда указывают относительно положения точки весеннего равноденствия, эклиптики (или экватора) на определенный момент времени, называемый эпохой. Для ссылки на этот факт используются выражения «эпоха 1950,0» или «равноденствие 2000,0». Эпоху обязательно нужно учитывать для точного выполнения преобразований координат. Чтобы получить значение наклона эклиптики на требуемую эпоху, пользуются следующей формулой:

$$\varepsilon = 23\overset{\circ}{.}43929111 - 46\overset{''}{.}8150T - 0\overset{''}{.}00059T^2 + 0\overset{''}{.}001813T^3, \tag{2.9}$$

здесь T — число юлианских столетий, отделяющих эпоху от полудня 1 января 2000 года. В отличие от григорианского календаря, в котором средняя продолжительность столетия составляет 36524,25 суток, юлианское столетие состоит ровно из 36 525 суток. Важнейшие юлианские эпохи — J1900 (0,5 января 1900, JD 2415020,0) и J2000 (1,5 января 1900, JD 2451545,0). Интервал между ними составляет ровно одно юлианское столетие. Величину T для данной эпохи можно вычислить по ее юлианской дате

$$T = (\text{JD } 2451545)/36525.$$

Однако в большинстве случаев хватает и меньшей точности. Поскольку за год значение ε изменяется меньше, чем на $0\overset{''}{.}5$, то хорошее приближение дает форму-

ла $T \approx (y - 2000)/100$, где y — год нужной вам эпохи. Функция Equ2EclMatrix определяет для заданной эпохи матрицу преобразования экваториальных координат в эклиптические.

```
//------------------------------------------------------------------------------
// Equ2EclMatrix: Преобразование экваториальных координат в эклиптические
//   T          Время в юлианских столетиях, истекшее с эпохи J2000
// <return>:   Матрица преобразования
//------------------------------------------------------------------------------
Mat3D Equ2EclMatrix (double T) {
  const double
    eps = ( 23.43929111-(46.8150+(0.00059-0.001813*T)*T)*T/3600.0 ) * Rad;
  return R_x(eps);
}
```

Матрицу обратного преобразования можно получить, используя функцию Ecl2EquMatrix.

```
//------------------------------------------------------------------------------
// Ecl2EquMatrix: Преобразование эклиптических координат в экваториальные
//   T          Время в юлианских столетиях, истекшее с эпохи J2000
// <return>:   Матрица преобразования
//------------------------------------------------------------------------------
Mat3D Ecl2EquMatrix (double T)
{
  return Transp(Equ2EclMatrix(T));
}
```

С применением этой функции рассмотренный выше пример будет выглядеть так (для эпохи J1950 = JD2433282,50):

```
double MJD = 33282.0;                  // Эпоха
double T   = (MJD-MJD.J2000)/36525.0;
Mat3D  U   = Ecl2EquMatrix(T);         // Матрица преобразования
Vec3D  e   = Vec3D(1.0,2.0,3.0);       // Эклиптические координаты
Vec3D  a   = U*e;                      // Экваториальные координаты
cout << "l   " << Deg*e[phi]      << " b   " << Deg*e[theta] << endl
     << "RA " << Deg*a[phi]/15.0 << " Dec " << Deg*a[theta] << endl;
```

2.4. Прецессия

Силы, действующие на Землю со стороны Солнца, Луны и планет, вызывают изменение взаиморасположения земной оси и эклиптики. Это изменение проявляется не только в медленном изменении угла ε, но и в довольно значительном изменении положения точки весеннего равноденствия. Она перемещается относительно звезд примерно на $1^\circ\!.5$ в столетие ($1'$ в год). Очевидно, что для точных вычислений изменение ее положения должно обязательно учитываться. Чаще всего используются следующие эпохи:

- эпоха текущей даты;
- эпоха J2000;
- эпоха B1950.

Использование эпохи текущей даты означает, что при вычислениях используются те положения экватора, эклиптики и точки весеннего равноденствия, которые имеют место на момент рассматриваемой даты. Учитывать такое ежедневное изменение системы координат нужно, например, при использовании координат планет для наведения на них телескопа, установленного на экваториальной монтировке. Поскольку положение земной оси в пространстве меняется, будет меняться и положение полярной оси телескопа. С другой стороны, для изучения реальных движений планет в пространстве лучше использовать фиксированную эпоху, например J2000 (1,5 января 2000 = JD 2451545,0), которую стали широко применять с 1984 года. Раньше в течение долгого времени использовалась эпоха B1950. Данные многих звездных каталогов и атласов отнесены к этой эпохе. В качестве примеров можно привести «SAO Star Catalog» и «Atlas Coeli». Префикс B указывает, что эта эпоха соответствует не середине интервала между эпохами J1900 и J2000 (которая, естественно, будет J1950 = JD 2433282,5), а началу 1950 бесселева года (0,923 января 1950 = JD 2433282,423).

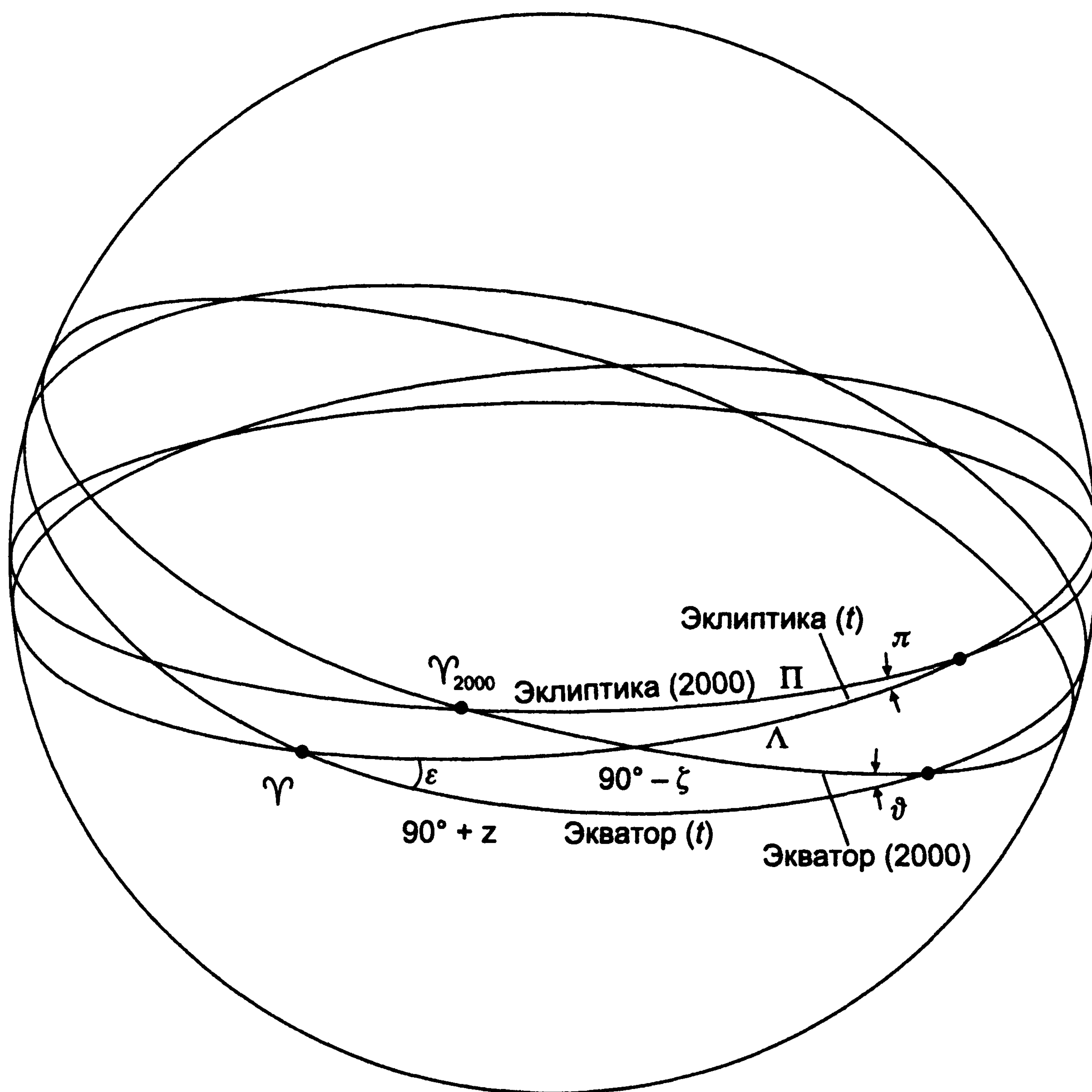


Рис. 2.4. Влияние прецессии на взаиморасположение эклиптики, экватора и точки весеннего равноденствия

Для преобразования координат от одной эпохи к другой требуются вычисления, аналогичные тем, что использовались для преобразования эклиптических

координат в экваториальные. Но если раньше системы координат различались лишь поворотом относительно оси x, то теперь мы имеем дело с тремя последовательными элементарными поворотами: сначала вокруг оси z, затем вокруг оси x и, наконец, снова вокруг оси z.

Если мы рассмотрим положения плоскости эклиптики в два момента времени T_0 и T_0+T, то угол между этими двумя плоскостями составит (рис. 2.4):

$$\pi = (47''\!.0029 - 0''\!.06603\cdot T_0 + 0''\!.000598\cdot T_0^2)\cdot T + \\ +(-0''\!.03302 + 0''\!.000598\cdot T_0)\cdot T^2 + 0''\!.000060\cdot T^3. \qquad (2.10)$$

Рассмотрим две системы координат (x',y',z') и (x'',y'',z''), у которых плоскости (x',y') и (x'',y'') совпадают с положениями плоскости эклиптики в моменты T_0 и T_0+T соответственно. Переход между ними выполняется по следующим формулам:

$$x'' = x',$$
$$y'' = +\cos\pi\cdot y' + \sin\pi\cdot z',$$
$$z'' = -\sin\pi\cdot y' + \cos\pi\cdot z'.$$

Здесь предполагается, что оси x' и x'' совпадают с прямой, по которой пересекаются рассматриваемые две плоскости. Пусть набор координат (x_0,y_0,z_0) задает положение точки на эпоху T_0, а (x,y,z) соответственно на эпоху T_0+T. Тогда получаем

$$x' = +\cos\Pi\cdot x_0 + \sin\Pi\cdot y_0,$$
$$y' = -\sin\Pi\cdot x_0 + \cos\Pi\cdot y_0,$$
$$z' = z_0$$

и

$$x = +\cos\Lambda\cdot x'' - \sin\Lambda\cdot y'',$$
$$y = +\sin\Lambda\cdot x'' + \cos\Lambda\cdot y'',$$
$$z = z'',$$

где

$$\Pi = (174^\circ\!.876383889 + 3289''\!.4789 T_0 + 0''\!.60622 T_0^2) + \\ +(-869''\!.8089 - 0''\!.50491 T_0)T + 0''\!.03536 T_2,$$
$$p = (5029''\!.0966 + 2''\!.22226 T_0 - 0''\!.000042 T_0^2)T + \\ +(1''\!.11113 - 0''\!.000042 T_0)T^2 - 0''\!.000006 T^3, \qquad (2.11)$$
$$\Lambda = \Pi + p.$$

Π — это угол между осью x' и направлением на точку весеннего равноденствия Υ_0 для эпохи T_0 (ось x_0), Λ — угол между x'' и точкой весеннего равноденствия эпохи T_0+T (ось x). p обозначает прецессию в долготе, поскольку перемещение

точки весеннего равноденствия происходит в основном по долготе. Выполняя необходимые подстановки, получаем уравнение

$$\boldsymbol{r} = \boldsymbol{P}\boldsymbol{r}_0, \text{ где } \boldsymbol{P} = \boldsymbol{R}_z(-\Pi - p)\boldsymbol{R}_x(\pi)\boldsymbol{R}_z(\Pi). \quad (2.12)$$

Запишем матрицу $\boldsymbol{P}$ поэлементно:

$$\begin{aligned}
p_{11} &= +\cos\Lambda\cos\Pi + \sin\Lambda\cos\pi\sin\Pi \\
p_{21} &= +\sin\Lambda\cos\Pi - \cos\Lambda\cos\pi\sin\Pi \\
p_{31} &= +\sin\pi\ \sin\Pi \\
p_{12} &= +\cos\Lambda\sin\Pi - \sin\Lambda\cos\pi\cos\Pi \\
p_{22} &= +\sin\Lambda\sin\Pi + \cos\Lambda\cos\pi\cos\Pi. \\
p_{32} &= -\sin\pi\cos\Pi \\
p_{13} &= -\sin\Lambda\sin\pi \\
p_{23} &= +\cos\Lambda\sin\pi \\
p_{33} &= +\cos\pi.
\end{aligned} \quad (2.13)$$

В экваториальных координатах углам π, Π и Λ соответствуют углы $90°-\zeta$, ϑ и $90° + z$:

$$\begin{aligned}
\zeta &= (2306''\!.2181 + 1''\!.39656T_0 - 0''\!.000139T_0^2)T + \\
&\quad + (0''\!.30188 - 0''\!.000345T_0)T^2 + 0''\!.017998T^3, \\
\vartheta &= (2004''\!.3109 - 0''\!.85330T_0 - 0''\!.000217\,T_0^2)T + \\
&\quad + (-0''\!.42665 - 0''\!.000217\,T_0)T^2 - 0''\!.041833T^3, \\
z &= \zeta + (0''\!.79280 + 0''\!.000411T_0)T^2 + 0''\!.000205\,T^3.
\end{aligned} \quad (2.14)$$

Отсюда получаем аналогичную матрицу

$$\boldsymbol{P} = \boldsymbol{R}_z(-z)\boldsymbol{R}_y(\vartheta)\boldsymbol{R}(-\zeta) \quad (2.15)$$

или, поэлементно

$$\begin{aligned}
p_{11} &= -\sin z\sin\zeta + \cos z\cos\vartheta\cos\zeta \\
p_{21} &= +\cos z\sin\zeta + \sin z\cos\vartheta\cos\zeta \\
p_{31} &= +\sin\vartheta\cos\zeta \\
p_{12} &= -\sin z\cos\zeta - \cos z\cos\vartheta\sin\zeta \\
p_{22} &= +\cos z\cos\zeta + \sin z\cos\vartheta\sin\zeta \\
p_{32} &= -\sin\vartheta\sin\zeta \\
p_{13} &= -\cos z\sin\vartheta \\
p_{23} &= -\sin z\sin\vartheta \\
p_{33} &= +\cos\vartheta.
\end{aligned} \quad (*2.16)$$

Наиболее трудоемкая операция при вычислении прецессии — это определение различных углов и компонентов p_{ij}. Однако значение матрицы преобразования зависит только от выбранных двух эпох T_0 и $T_0 + T$, и ее можно многократно использовать для перевода координат различных объектов между этими эпохами.

```
//------------------------------------------------------------------------
// PrecMatrix_Ecl: Прецессия в эклиптических координатах
//   T1          Исходная эпоха
//   T2          Требуемая эпоха
// <return>:     Матрица преобразования координат
// Замечание: T1 и T2 отсчитываются в юлианских столетиях от J2000
//------------------------------------------------------------------------
Mat3D PrecMatrix_Ecl (double T1, double T2)
{
  const double dT = T2-T1;
  double Pi, pi, p_a;
  Pi  = 174.876383889*Rad +
        ( ((3289.4789+0.60622*T1)*T1) +
              ((-869.8089-0.50491*T1) + 0.03536*dT)*dT )/Arcs;
  pi  = ( (47.0029-(0.06603-0.000598*T1)*T1)+
              ((-0.03302+0.000598*T1)+0.000060*dT)*dT )*dT/Arcs;
  p_a = ( (5029.0966+(2.22226-0.000042*T1)*T1)+
              ((1.11113-0.000042*T1)-0.000006*dT)*dT )*dT/Arcs;
  return R_z(-(Pi+p_a)) * R_x(pi) * R_z(Pi);
}

//------------------------------------------------------------------------
// PrecMatrix_Equ: Прецессия в экваториальных координатах
//   T1          Исходная эпоха
//   T2          Требуемая эпоха
// <return>:     Матрица преобразования координат
// Замечание: T1 и T2 отсчитываются в юлианских столетиях от J2000
//------------------------------------------------------------------------
Mat3D PrecMatrix_Equ (double T1, double T2)
{
  const double dT = T2-T1;
  double zeta,z,theta;
  zeta  =  ( (2306.2181+(1.39656-0.000139*T1)*T1)+
               ((0.30188-0.000344*T1)+0.017998*dT)*dT )*dT/Arcs;
  z     =  zeta + ( (0.79280+0.000411*T1)+0.000205*dT)*dT*dT/Arcs;
  theta =  ( (2004.3109-(0.85330+0.000217*T1)*T1)-
               ((0.42665+0.000217*T1)+0.041833*dT)*dT )*dT/Arcs;
  return R_z(-z) * R_y(theta) * R_z(-zeta);
}
```

Чтобы лучше разобраться в явлении прецессии и немного поупражняться в использовании приведенных здесь функций, рассмотрим еще один пример. Требуется преобразовать набор эклиптических координат, заданных на эпоху B1950, в экваториальные координаты на эпоху J2000. Это можно сделать двумя способами, в зависимости от того, с чего начать: с учета прецессии или с преобразования эклиптических координат к экваториальным.

```
// Объявления
double T_B1950 = -0.500002108;          // Epoch B1950 (JD2433282.423)
double l       = 200.0*Rad;             // Longitude
```

```
double b   =  10.0*Rad;                   // Latitude
Vec3D  r_0 = Vec3D(Polar(l,b));           // Ecliptic coordinates B1950
Vec3D  r;

// Первый метод
r = PrecMatrixEcl(T_B1950,T_J2000) * r_0;  // Ecliptic J2000
cout << "l,b (J2000)     " << Deg*r[phi]      << Deg*r[theta] << endl;
r = Ecl2EquMatrix(T_J2000) * r;            // Equator J2000
cout << "RA,Dec (J2000)  " << Deg*r[phi]/15.0 << Deg*r[theta] << endl;

// Второй метод
r = Ecl2EquMatrix(T_B1950) * r_0;          // Equator B1950
cout << "RA,Dec (B1950)  " << Deg*r[phi]/15.0 << Deg*r[theta] << endl;
r = PrecMatrixEqu(T_B1950,T_J2000) * r;    // Equator J2000
cout << "RA,Dec (J2000) " << Deg*r[phi]/15.0 << Deg*r [theta] << endl;
```

2.5. Геоцентрические координаты и солнечная орбита

Все, что теперь нам требуется для придания программе законченного вида, — это метод для перехода от гелиоцентрических координат (отнесенных к центру Солнца) к геоцентрическим (отнесенных к центру Земли). Перенос начала координат описывается уравнениями

$$\boldsymbol{r}_p = \boldsymbol{r}_{\oplus p} + \boldsymbol{r}_{\odot} \quad \text{и} \quad \boldsymbol{r}_{\oplus p} = \boldsymbol{r}_p - \boldsymbol{r}_{\odot}, \tag{2.17}$$

которые легко получить на основе треугольника Солнце–Земля–планета, приведенного на рис. 2.5.

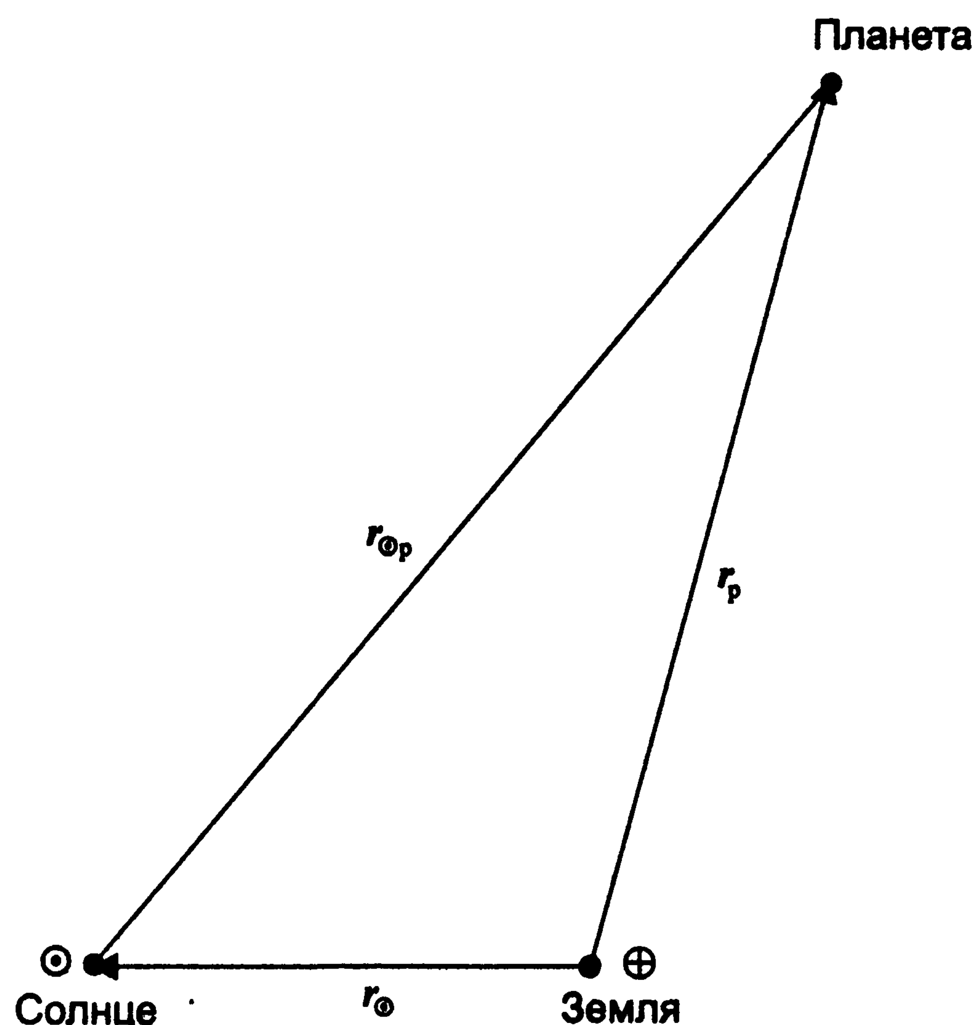


Рис. 2.5. Треугольник Земля–Солнце–планета

Здесь $\boldsymbol{r}_{\odot p}$ и $\boldsymbol{r}_p$ — соответственно гелиоцентрический и геоцентрический радиус-векторы точки P, а $\boldsymbol{r}_\odot$ — геоцентрический радиус-вектор Солнца. Записав эти уравнения покомпонентно, получаем

$$\begin{aligned} x_p &= x_{\odot p} + x_\odot, & \qquad x_{\odot p} &= x_p - x_\odot, \\ y_p &= y_{\odot p} + y_\odot, & \text{и} \qquad y_{\odot p} &= y_p - z_\odot, \\ z_p &= z_{\odot p} + z_\odot & \qquad z_{\odot p} &= z_p - z_\odot. \end{aligned}$$

Здесь безразлично, какие используются координаты — эклиптические или экваториальные.

Поскольку любая погрешность в определении положения Солнца неизбежно сказывается на точности преобразования, постараемся вычислить его координаты по возможности точно. Приводимая здесь функция SunPos обеспечивает точность около 1″. Для большинства задач этого достаточно.

Для эпохи

$$T = (JD - 2451545)/36525$$

декартовы эклиптические координаты Солнца

$$\boldsymbol{r}_\odot = \begin{pmatrix} x_\odot \\ y_\odot \\ z_\odot \end{pmatrix} = \begin{pmatrix} R\cos B\cos L \\ R\cos B\sin L \\ R\sin B \end{pmatrix}. \qquad (2.18)$$

Как всегда, эклиптические долгота L и широта B, вычисленные первоначально для одной эпохи, могут быть приведены к другой эпохе при помощи функций, которые мы уже обсуждали выше.

```
//------------------------------------------------------------------------------
//
// SunPos: Вычисляет эклиптические координаты Солнца на основе разложений
//         в аналитические ряды
//   T         Эпоха, выраженная в юлианских столетиях с момента J2000
//
// <return>:   Геоцентрическое положение Солнца (в [a.e.]), отнесенное
//             к эклиптике и равноденствию указанной эпохи
//
//------------------------------------------------------------------------------
Vec3D SunPos (double T)
{
  // Variables
  double  M2,M3,M4,M5,M6;           // Средние аномалии
  double  D, A, U;                  // Средние аргументы лунной орбиты
  Pert    Ven, Mar, Jup, Sat;       // Возмущения
  double  dl, dr, db;               // Поправки в долготе ["],
                                    // радиусе [a.e.] и широте ["]
  double  l,b,r;                    // Эклиптические координаты

  // Средние аномалии планет и средние аргументы лунной орбиты [рад]
  M2 = pi2 * Frac ( 0.1387306 + 162.5485917*T );
  M3 = pi2 * Frac ( 0.9931266 +  99.9973604*T );
  M4 = pi2 * Frac ( 0.0543250 +  53.1666028*T );
```

```
M5 = pi2 * Frac ( 0.0551750 +   8.4293972*T );
M6 = pi2 * Frac ( 0.8816500 +   3.3938722*T );

D  = pi2 * Frac ( 0.8274 + 1236.8531*T );
A  = pi2 * Frac ( 0.3749 + 1325.5524*T );
U  = pi2 * Frac ( 0.2591 + 1342.2278*T );

// Кеплеровские члены и возмущения со стороны Венеры
Ven.Init ( T, M3,0,7, M2,-6,0 );
Ven.Term ( 1, 0,0,-0.22,6892.76,-16707.37, -0.54, 0.00, 0.00);
Ven.Term ( 1, 0,1,-0.06, -17.35,    42.04, -0.15, 0.00, 0.00);
Ven.Term ( 1, 0,2,-0.01,  -0.05,     0.13, -0.02, 0.00, 0.00);
Ven.Term ( 2, 0,0, 0.00,  71.98,  -139.57,  0.00, 0.00, 0.00);
Ven.Term ( 2, 0,1, 0.00,  -0.36,     0.70,  0.00, 0.00, 0.00);
Ven.Term ( 3, 0,0, 0.00,   1.04,    -1.75,  0.00, 0.00, 0.00);
Ven.Term ( 0,-1,0, 0.03,  -0.07,    -0.16, -0.07, 0.02,-0.02);
Ven.Term ( 1,-1,0, 2.35,  -4.23,    -4.75, -2.64, 0.00, 0.00);
Ven.Term ( 1,-2,0,-0.10,   0.06,     0.12,  0.20, 0.02, 0.00);
Ven.Term ( 2,-1,0,-0.06,  -0.03,     0.20, -0.01, 0.01,-0.09);
Ven.Term ( 2,-2,0,-4.70,   2.90,     8.28, 13.42, 0.01,-0.01);
Ven.Term ( 3,-2,0, 1.80,  -1.74,    -1.44, -1.57, 0.04,-0.06);
Ven.Term ( 3,-3,0,-0.67,   0.03,     0.11,  2.43, 0.01, 0.00);
Ven.Term ( 4,-2,0, 0.03,  -0.03,     0.10,  0.09, 0.01,-0.01);
Ven.Term ( 4,-3,0, 1.51,  -0.40,    -0.88, -3.36, 0.18,-0.10);
Ven.Term ( 4,-4,0,-0.19,  -0.09,    -0.38,  0.77, 0.00, 0.00);
Ven.Term ( 5,-3,0, 0.76,  -0.68,     0.30,  0.37, 0.01, 0.00);
Ven Term ( 5,-4,0,-0.14,  -0.04,    -0.11,  0.43,-0.03, 0.00);
Ven.Term ( 5,-5,0,-0.05,  -0.07,    -0.31,  0.21, 0.00, 0.00);
Ven.Term ( 6,-4,0, 0.15,  -0.04,    -0.06, -0.21, 0.01, 0.00);
Ven.Term ( 6,-5,0,-0.03,  -0.03,    -0.09,  0.09,-0.01, 0.00);
Ven.Term ( 6,-6,0, 0.00,  -0.04,    -0.18,  0.02, 0.00, 0.00);
Ven.Term ( 7,-5,0,-0.12,  -0.03,    -0.08,  0.31,-0.02,-0.01);
dl = Ven.dl(); dr = Ven.dr();  db = Ven.db();

// Возмущения со стороны Марса
Mar.Init ( T, M3,1,5, M4,-8,-1 );
Mar Term ( 1,-1,0,-0.22,   0.17,    -0.21, -0.27, 0.00, 0.00);
Mar.Term ( 1,-2,0,-1.66,   0.62,     0.16,  0.28, 0.00, 0.00);
Mar.Term ( 2,-2,0, 1.96,   0.57,    -1.32,  4.55, 0.00, 0.01);
Mar.Term ( 2,-3,0, 0.40,   0.15,    -0.17,  0.46, 0.00, 0.00);
Mar.Term ( 2,-4,0, 0.53,   0.26,     0.09, -0.22, 0.00, 0.00);
Mar.Term ( 3,-3,0, 0.05,   0.12,    -0.35,  0.15, 0.00, 0.00),
Mar.Term ( 3,-4,0,-0.13,  -0.48,     1.06, -0.29, 0.01, 0.00);
Mar.Term ( 3,-5,0,-0.04,  -0.20,     0.20, -0.04, 0.00, 0.00);
Mar.Term ( 4,-4,0, 0.00,  -0.03,     0.10,  0.04, 0.00, 0.00);
Mar.Term ( 4,-5,0, 0.05,  -0.07,     0.20,  0.14, 0.00, 0.00);
Mar.Term ( 4,-6,0,-0.10,   0.11,    -0.23, -0.22, 0.00, 0.00);
Mar.Term ( 5,-7,0,-0.05,   0.00,     0.01, -0.14, 0.00, 0 00);
Mar.Term ( 5,-8,0, 0.05,   0.01,    -0.02,  0.10, 0.00, 0.00);
dl += Mar.dl(), dr += Mar.dr();  db += Mar.db();

// Возмущения со стороны Юпитера
Jup.Init ( T, M3,-1,3, M5,-4,-1 );
Jup.Term (-1,-1,0, 0.01,   0.07,     0.18, -0.02, 0.00,-0.02);
Jup.Term ( 0,-1,0,-0.31,   2.58,     0.52,  0.34, 0.02, 0.00);
Jup.Term ( 1,-1,0,-7.21,  -0.06,     0.13,-16.27, 0.00,-0.02);
Jup.Term ( 1,-2,0,-0.54,  -1.52,     3.09, -1.12, 0.01,-0.17);
Jup.Term ( 1,-3,0,-0.03,  -0.21,     0.38, -0.06, 0.00,-0.02);
```

```
  Jup.Term ( 2,-1,0,-0.16,  0.05,   -0.18, -0.31, 0.01, 0.00);
  Jup.Term ( 2,-2,0, 0.14, -2.73,    9.23,  0.48, 0.00, 0.00);
  Jup.Term ( 2,-3,0, 0.07, -0.55,    1.83,  0.25, 0.01, 0.00);
  Jup.Term ( 2,-4,0, 0.02, -0.08,    0.25,  0.06, 0.00, 0.00);
  Jup.Term ( 3,-2,0, 0.01, -0.07,    0.16,  0.04, 0.00, 0.00);
  Jup.Term ( 3,-3,0,-0.16, -0.03,    0.08, -0.64, 0.00, 0.00);
  Jup.Term ( 3,-4,0,-0.04, -0.01,    0.03, -0.17, 0.00, 0.00);
  dl += Jup.dl(); dr += Jup.dr();  db += Jup.db();

  // Возмущения со стороны Сатурна
  Sat.Init ( T, M3,0,2, M6,-2,-1 );
  Sat.Term ( 0,-1,0, 0.00,  0.32,    0.01,  0.00, 0.00, 0.00);
  Sat.Term ( 1,-1,0,-0.08, -0.41,    0.97, -0.18, 0.00,-0.01);
  Sat.Term ( 1,-2,0, 0.04,  0.10,   -0.23,  0.10, 0.00, 0.00);
  Sat.Term ( 2,-2,0, 0.04,  0.10,   -0.35,  0.13, 0.00, 0.00);
  dl += Sat.dl(); dr += Sat.dr();  db += Sat.db();

  // Учет различия между барицентром системы Земля-Луна и центром Земли
  dl += +  6.45*sin(D) - 0.42*sin(D-A) + 0.18*sin(D+A)
        +  0.17*sin(D-M3) - 0.06*sin(D+M3);
  dr += + 30.76*cos(D) - 3.06*cos(D-A) + 0.85*cos(D+A)
        -  0.58*cos(D+M3) + 0.57*cos(D-M3);
  db += + 0.576*sin(U);

  // Долгопериодические возмущения
  dl += + 6.40 * sin ( pi2*(0.6983 + 0.0561*T) )
        + 1.87 * sin ( pi2*(0.5764 + 0.4174*T) )
        + 0.27 * sin ( pi2*(0.4189 + 0.3306*T) )
        + 0.20 * sin ( pi2*(0.3581 + 2.4814*T) );

  // Эклиптические координаты ([рад],[а.е.])
  l = pi2 * Frac ( 0.7859453 + M3/pi2 +
                 ( (6191.2+1.1*T)*T + dl ) / 1296.0e3 );
  r = 1.0001398 - 0.0000007 * T + dr * 1.0e-6;
  b = db / Arcs;

  return Vec3D ( Polar(l,b,r) );  // Радиус-вектор
}
```

Большой размер функции SunPos объясняется тем, что относительное движение Солнца и Земли нельзя описать с необходимой точностью простой эллиптической орбитой. Помимо возмущений со стороны других планет — среди которых особое значение имеют Венера и Юпитер, — надо также учитывать колебания центра Земли относительно барицентра системы Земля–Луна, происходящие с периодом около месяца. Строго говоря, в плоскости эклиптики движется не сама Земля, а именно барицентр. Поэтому движение Луны вызывает небольшие периодические возмущения геоцентрической орбиты Солнца. Мы не будем здесь более подробно рассматривать эту функцию, а вернемся к ней снова в главе 6, после рассмотрения соответствующих процедур для планет.

2.6. Программа Coco

Теперь пришло время объединить описанные функции в готовую программу. Далее приводятся все необходимые процедуры и главная программа. Рассмот-

ренные нами функции собраны в библиотечные модули, заголовки которых подключаются в начале программы.

Основные функции программы включены в класс Position, который наряду с операциями ввода и вывода имеет три метода для выбора начала координат (гелиоцентрические или геоцентрические), эпохи и системы координат (эклиптическая или экваториальная). Главная программа, таким образом, ограничивается объявлением объекта класса Position и вызовами его методов по выбору пользователя. Хотя класс Position не используется за пределами программы Coco, применение здесь объектно-ориентированного подхода обеспечивает компактный и ясный стиль программирования.

```
//-------------------------------------------------------------------------
// Файл:        Coco.cpp
// Назначение: Преобразования координат
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//-------------------------------------------------------------------------

#include <iomanip>
#include <iostream>
#include "APC_Const.h"
#include "APC_Math.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"
using namespace std;

// Определение класса Position
enum enOrigin   { Heliocentric, Geocentric };  // Начало координат
enum enRefSys   { Ecliptic, Equator };         // Система координат
class Position
{
  public:
    void Input(); // Запрос параметров у пользователя
    void SetOrigin(enOrigin Origin);
    void SetRefSys(enRefSys RefSys);
    void SetEquinox(double T_Equinox);
    void Print();
  private:
    Vec3D      m_R;           // Радиус-вектор
    enOrigin   m_Origin;      // Начало координат
    enRefSys   m_RefSys;      // Система координат
    double     m_TEquinox;    // Равноденствие (в столетиях от J2000)
    double     m_MjdEpoch;    // Эпоха (модифицированная юлианская дата)
};

// Ввод и вывод данных
void Position::Input() { ... }
void Position::Print() { ... }
// Изменение начала координат
void Position::SetOrigin(enOrigin Origin)
{
  double T_Epoch;
```

```
  Vec3D  R_Sun,
  if (Origin!=m_Origin) {
    // Геоцентрические координаты Солнца на указанную эпоху,
    // отнесенные к заданной системе координат и равноденствию
    T_Epoch = (m_MjdEpoch-MJD_J2000)/36525.0;
    if (m_RefSys==Ecliptic)
      R_Sun = PrecMatrix_Ecl(T_Epoch,m_TEquinox) * SunPos(T_Epoch),
    else
      R_Sun = Ecl2EquMatrix(m_TEquinox) *
              PrecMatrix_Ecl(T_Epoch,m_TEquinox) * SunPos(T_Epoch);
    // Изменение начала координат
    if (m_Origin==Heliocentric) {
      m_R += R_Sun; m_Origin = Geocentric;
    }
    else {
      m_R -= R_Sun; m_Origin = Heliocentric;
    };
  };
}
// Изменение системы координат
void Position::SetRefSys(enRefSys RefSys)
{
  if (RefSys!=m_RefSys) {
    if (m_RefSys==Equator) {
      m_R = Equ2EclMatrix(m_TEquinox) * m_R;  m_RefSys = Ecliptic;
    }
    else {
      m_R = Ecl2EquMatrix(m_TEquinox) * m_R;  m_RefSys = Equator;
    }
  };
}
// Изменение равноденствия
void Position::SetEquinox(double T_Equinox)
{
  if (T_Equinox!=m_TEquinox) {
    if (m_RefSys==Equator)
      m_R = PrecMatrix_Equ(m_TEquinox,T_Equinox) * m_R;
    else
      m_R = PrecMatrix_Ecl(m_TEquinox,T_Equinox) * m_R;
    m_TEquinox = T_Equinox;
  };
}
//------------------------------------------------------------------------
//
// Основная программа
//
//------------------------------------------------------------------------
void main() {
  // Variables
  Position  Pos;
  char      c;
  bool      End = false;
  double    Year;

  // Вывод заголовка
  cout << endl
       << "                    COCO: coordinate conversions        " << endl
```

```
      << "       (c) 1999 Oliver Montenbruck, Thomas Pfleger" << endl
      << endl;

  // Инициализация
  Pos.Input();
  Pos.Print();

  // Цикл обработки команд
  do {
    // Ввод команды
    cout << "Enter command (?=Help) ... ";
    cin >> c; cin.ignore(81,'\n'); c=tolower(c);

    // Действие
    switch (c) {
      case 'x':
        End = true;  break;
      case 'a':
        Pos.SetRefSys(Equator); Pos.Print();  break;
      case 'e':
        Pos.SetRefSys(Ecliptic); Pos.Print();  break;
      case 'g':
        Pos.SetOrigin(Geocentric); Pos.Print();  break;
      case 'h':
        Pos.SetOrigin(Heliocentric); Pos.Print();  break;
      case 'n':
        Pos.Input(); Pos.Print();  break;
      case 'p':
        cout << "New equinox (yyyy.y)   ... ";
        cin >> Year; cin.ignore(81,'\n');
        Pos.SetEquinox( (Year-2000.0)/100.0 );
        Pos.Print();
        break;
      default:
        // Отображение подсказки
        cout << endl
             << "Available commands" << endl
             << "  e=ecliptic,   a=equatorial,   p=precession,  " << endl
             << "  g=geocentric, h=heliocentric, n=new input,   " << endl
             << "  x=exit                                       " << endl
             << endl;
        break;
    }
  }
  while (!End);
  cout << endl;
}
```

Следующий пример демонстрирует возможности программы Coco. Координаты, подлежащие преобразованию, могут вводиться как в экваториальной, так и в эклиптической системах. Можно выбирать между прямоугольными и полярными координатами. Текст, вводимый пользователем, выделен курсивом. В качестве примера мы выбрали экваториальные координаты точки весеннего равноденствия, заданные в полярной форме на эпоху 1950,0. Расстояние произвольно задано равным 1 а. е., оно, конечно, всегда должно быть положительным.

```
            COCO: coordinate conversions
      (c) 1999 Oliver Montenbruck, Thomas Pfleger

New input:

  Reference system (e=ecliptic,a=equator) ... a
  Format (c=cartesian,p=polar)            ... p
  Coordinates (RA [h m s] Dec [o ' "] R)  ... 0 0 0.0 0 0 0.0 1.0
  Equinox (yyyy.y)                        ... 1950.0
  Origin (h=heliocentric,g=geocentric)    ... g
  Epoch (yyyy mm dd hh.h)                 ... 1989 1 1 0.0
```

Далее программа Coco выполняет контрольный вывод данных в прямоугольных и полярных координатах:

```
Geocentric equatorial coordinates
(Equinox J1950.0,  Epoch 1989/01/01 00.0)

  (x,y,z) = (  1.00000000,  0.00000000,  0.00000000)

        h  m  s              o  '  "
  RA =  0 00 00.00    Dec = + 0 00 00.0    R =   1.00000000


Enter command (?=Help) ... ?

Available commands
  e=ecliptic,   a=equatorial,   p=precession,
  g=geocentric, h=heliocentric, n=new input,
  x=exit

Enter command (?=Help) ...
```

Теперь программа ожидает ввода команды преобразования координат. Имеются следующие варианты:

- a преобразование в экваториальные координаты;
- e преобразование в эклиптические координаты;
- p прецессия (выбор эпохи);
- g преобразование в геоцентрические координаты;
- h преобразование в гелиоцентрические координаты;
- ? подсказка;
- x выход (завершение программы).

Прежде всего мы хотим привести координаты заданной точки к эпохе 2000,0 и поэтому выберем вариант p, чтобы рассчитать прецессию. Введя нужный год, мы получаем координаты точки весеннего равноденствия 1950 года, отнесенные к новой эпохе 2000,0. Прецессия составляет чуть меньше половины градуса.

```
Enter command (?=Help) ... p
New equinox (yyyy.y)   ... 2000.0

Geocentric equatorial coordinates
(Equinox J2000.0,  Epoch 1989/01/01 00.0)
```

```
  (x,y,z) = (  0.99992571,  0.01117889,  0.00485898)

        h  m  s                o  '  "
  RA =  0 02 33.73     Dec = + 0 16 42.2    R =   1.00000000
```

Теперь нам нужно преобразовать введенные ранее экваториальные координаты в эклиптические. Для этого выберем команду e и получим преобразованные координаты в прямоугольной и полярной формах. Вместо прямого восхождения RA и склонения Dec мы получили эклиптическую долготу L и широту B. Как и на предыдущем шаге, расстояние не меняется.

```
Enter command (?=Help) ... e

Geocentric ecliptic coordinates
(Equinox J2000.0, Epoch 1989/01/01 00.0)

  (x,y,z) = (  0.99992571,  0.01218922,  0.00001132)

        o  '  "              o  '  "
  L =   0 41 54.27     B = + 0 00 02.3    R =   1.00000000
```

Теперь можно преобразовать полученные координаты из геоцентрических в гелиоцентрические на заданную эпоху, используя для этого команду h. Поскольку на предыдущем шаге мы выбрали эклиптические координаты, теперь мы получим гелиоцентрические эклиптические координаты.

```
Enter command (?=Help) ... h

Heliocentric ecliptic coordinates
(Equinox J2000.0, Epoch 1989/01/01 00.0)

  (x,y,z) = (  0.81725247,  0.97838164,  0.00003597)

        o  '  "              o  '  "
  L =  50 07 39.50     B = + 0 00 05.8    R =   1.27480674
```

Преобразование в эклиптические координаты можно отменить, выбрав команду a, которая преобразует текущие (эклиптические) координаты в экваториальные.

```
Enter command (?=Help) ... a

Heliocentric equatorial coordinates
(Equinox J2000.0, Epoch 1989/01/01 00.0)

  (x,y,z) = (  0.81725247,  0.89763329,  0.38921086)

        h  m  s                o  '  "
  RA =  3 10 44.07     Dec = +17 46 36.5    R =   1.27480674
```

Теперь давайте пересчитаем координаты к исходной эпохе 1950,0.

```
Enter command (?=Help) ... p
New equinox (yyyy.y)   ... 1950.0

Heliocentric equatorial coordinates
(Equinox J1950.0, Epoch 1989/01/01 00.0)
```

```
  (x,y,z) = (  0.82911747,  0.88843066,  0.38521087)

         h  m  s                o  '  "
  RA =  3 07 54.68    Dec = +17 35 17.2    R =   1.27480674
```

Преобразовав эти координаты в геоцентрические, мы должны получить исходные координаты точки весеннего равноденствия, поскольку получится, что мы выполнили три преобразования координат, обратные трем первоначальным. Тем не менее неизбежные при компьютерных вычислениях ошибки округления могут привести к тому, что полученный результат не будет абсолютно точно совпадать с исходными значениями.

```
Enter command (?=Help) ... g

Geocentric equatorial coordinates
(Equinox J1950.0, Epoch 1989/01/01 00.0)

  (x,y,z) = (  1.00000000, -0.00000000,  0.00000000)

         h  m  s                o  '  "
  RA = 24 00 00.00    Dec = + 0 00 00.0    R =   1.00000000


Enter command (?=Help) ... x
```

Приведенные здесь значения могут слегка различаться в зависимости от используемого компилятора C++.

3. Вычисление моментов восхода и захода

3.1. Горизонтальная система координат наблюдателя

Рассмотренные нами эклиптические и экваториальные координаты определялись относительно средней плоскости земной орбиты и оси вращения Земли. К сожалению, ни одна из этих систем не подходит для наблюдателя, находящегося на поверхности Земли. Поскольку наш наблюдатель принимает участие во вращении Земли, то независимо от того, что он об этом думает, ему будет казаться, что Солнце, Луна и звезды, прочерчивая огромные дуги, двигаются по небу с востока на запад, достигая высшей точки своего пути в момент пересечения меридиана.

Видимые пути звезд зависят от географической широты точки наблюдения. Например, в южном полушарии большинство звезд достигает наибольшей высоты над горизонтом не на севере, а на юге. Поэтому наблюдателю, привыкшему видеть над собой звездное небо средних северных широт, трудно сориентироваться под южным небом — все известные созвездия выглядят как бы перевернутыми верх ногами.

На небесной сфере есть две очень важные точки: это зенит (точка, находящаяся прямо над головой наблюдателя) и северный полюс мира (рис. 3.1). Последний представляет собой точку, на которую указывает земная ось и вокруг которой, следовательно, происходит суточное вращение всех небесных светил. Высота этой точки над горизонтом соответствует географической широте наблюдателя φ. Большой круг, проходящий через северный полюс мира и зенит, называется меридианом. Он пересекает горизонт в точках севера и юга.

Используемые в горизонтальной системе координаты называются азимутом (A) и высотой (h). Их можно измерить, например, при помощи теодолита (рис. 3.1). Высота светила — это просто угол между ним и горизонтом. Азимут определяется углом поворота теодолита вокруг вертикальной оси, когда он переводится с точки юга в интересующее нас положение. С учетом этого азимут точки запада будет $A = 90°$, а точки востока соответственно $A = 270°$ (или $A = -90°$). Определенные неудобства создает тот факт, что в некоторых случаях, особенно в навигации, азимуты принято отсчитывать от точки севера. При таком выборе начала отсчета азимут точки востока будет составлять 90°. При малейшем сомнении совершенно необходимо точно выяснить, какое из двух определений азимута используется!

Вследствие вращения Земли высота и азимут объекта с заданными экваториальными координатами постоянно изменяются. Если зафиксировать положение телескопа в горизонтальной системе координат, то через его поле зрения будут проходить звезды с одинаковыми склонениями, но разными прямыми восхождениями. Таким образом, определение склонения звезды по ее видимой высоте и азимуту не зависит от времени. Что же касается прямых восхождений, то можно определить только разность между прямым восхождением интересующего нас объекта и звезды, которая в этот момент пересекает небесный меридиан.



Рис. 3.1. Горизонтальная система координат

Для того чтобы получить возможность свободно преобразовывать горизонтальные координаты в экваториальные и обратно, введем понятие *часового угла* τ. Он определяется как разность прямых восхождений наблюдаемого объекта и звезды на небесном меридиане. Как и прямое восхождение α, часовой угол обычно измеряется в единицах времени ($1^h \equiv 15°$), и поэтому примерно[1] соответствует времени, прошедшему с момента пересечения объектом меридиана. Направление отсчета часовых углов τ совпадает с направлением отсчета азимутов *A*. Используя декартовы координаты, можно записать

$$\boldsymbol{r} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \cos h \cos A \\ \cos h \sin A \\ \sin h \end{pmatrix}, \quad \hat{\boldsymbol{r}} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{pmatrix} = \begin{pmatrix} \cos\delta \cos\tau \\ \cos\delta \sin\tau \\ \sin\delta \end{pmatrix}. \tag{3.1}$$

Отсюда получаем преобразования:

$$\begin{aligned} x &= +\hat{x}\sin\varphi - \hat{z}\cos\varphi, & \hat{x} &= +x\sin\varphi + z\cos\varphi, \\ y &= +\hat{y}, & \hat{y} &= +y, \\ z &= +\hat{x}\cos\varphi + \hat{z}\sin\varphi, & \hat{z} &= -x\cos\varphi + z\sin\varphi, \end{aligned} \tag{3.2}$$

[1] Один оборот Земли вокруг своей оси укладывается в 23^h56^m.

В сокращенной форме их можно записать $\boldsymbol{r} = \boldsymbol{R}_y(\pi/2-\varphi)\hat{\boldsymbol{r}}$. Здесь φ — географическая широта наблюдателя, а значок ^ обозначает экваториальные координаты. Выполняя подстановку, получаем систему из трех уравнений:

$$\begin{aligned} \cos h\cos A &= +\cos\delta\cos\tau\sin\varphi-\sin\delta\cos\varphi \\ \cos h\sin A &= +\cos\delta\sin\tau \\ \sin h &= +\cos\delta\cos\tau\cos\varphi+\sin\delta\sin\varphi, \end{aligned} \tag{3.3}$$

позволяющих вычислить азимут и высоту по заданным часовому углу и склонению. Для обратного преобразования можно использовать следующие формулы:

$$\begin{aligned} \cos\delta\cos\tau &= +\cos h\cos A\sin\varphi+\sin h\cos\varphi \\ \cos\delta\sin\tau &= +\cos h\sin A \\ \sin\delta &= -\cos h\cos A\cos\varphi+\sin h\sin\varphi, \end{aligned} \tag{3.4}$$

позволяющие найти склонение и часовой угол по высоте и азимуту. Для преобразования координат между экваториальной и горизонтальной системами служат две функции — Equ2Hor и Hor2Equ. В них нет ничего оригинального, но мы воспроизводим их здесь ради полноты изложения.

```
//------------------------------------------------------------------------
// Equ2Hor: Преобразование экваториальных координат в горизонтальные
//   Dec      Склонение [рад]
//   tau      Часовой угол [рад]
//   lat      Географическая широта наблюдателя [рад]
//   h, Az    Высота и азимут [рад]
//------------------------------------------------------------------------
void Equ2Hor ( double Dec, double tau, double lat,
               double& h, double& Az )
{
  Vec3D e_equ, e_hor;
  e_equ = Vec3D(Polar(tau,Dec));          // единичный вектор в горизонтальной системе
  e_hor = R_y(pi/2.0-lat) * e_equ;        // единичный вектор в экваториальной системе
  Az = e_hor[phi];                        // полярный угол
  h  = e_hor[theta];
}
//------------------------------------------------------------------------
// Hor2Equ: Преобразование горизонтальных координат в экваториальные
//   h,Az     Высота и азимут [рад]
//   lat      Географическая широта наблюдателя [рад]
//   Dec      Склонение [рад]
//   tau      Часовой угол [рад]
//------------------------------------------------------------------------
void Hor2Equ ( double h, double Az, double lat,
               double& Dec, double& tau )
{
  Vec3D e_equ, e_hor;
  e_hor = Vec3D(Polar(Az,h));             // единичный вектор в горизонтальной системе
  e_equ = R_y(-(pi/2.0-lat)) * e_hor;     // единичный вектор в экваториальной системе
  tau = e_equ[phi];                       // полярный угол
  Dec = e_equ[theta];
}
```

Итак, мы научились преобразовывать координаты между горизонтальной и экваториальной системами. Однако для целей данной главы, то есть для определения моментов восхода и захода, нам понадобится только одно из полученных уравнений, а именно

$$\sin h = \cos\varphi\cos\delta\cos\tau + \sin\varphi\sin\delta. \quad (3.5)$$

Оно позволяет определить высоту над горизонтом по заданным значениям географической широты φ, склонения δ и часового угла τ.

Правда, чтобы воспользоваться этой формулой, нам нужно произвести некоторые приготовления. Прежде всего нам следует научиться определять экваториальные координаты (α, δ) Луны и Солнца. Затем понадобится уточнить, как, зная прямое восхождение светила, определить его часовой угол в заданный момент времени. Наконец, нам нужно будет учесть целый ряд поправок, которые влияют на высоту видимого горизонта. Только после выполнения всех этих шагов мы вернемся к только что приведенному уравнению.

3.2. Солнце и Луна

Для вычисления моментов восходов и заходов нет необходимости с высокой точностью определять координаты Солнца и Луны. Поэтому функции MiniSun и MiniMoon учитывают только самые важные члены уравнений, описывающих соответствующие орбиты. Они представляют собой значительно сокращенные версии программ SunPos и MoonPos, которые подробно описываются в главах 6 и 8. Для удобства в сокращенные функции также включено преобразование эклиптических координат в экваториальные, так что ими можно пользоваться для непосредственного получения прямого восхождения и склонения Солнца и Луны.

```
//------------------------------------------------------------------------------
// MiniMoon: Вычисляет экваториальные координаты Луны,
//           используя ограниченные по точности ряды
//   T         Время в юлианских столетиях от эпохи J2000
//   RA        Прямое восхождение Луны [рад]
//   Dec       Склонение Луны [рад]
//------------------------------------------------------------------------------
void MiniMoon (double T, double& RA, double& Dec)
{
  const double eps = 23.43929111*Rad;
  double L_0, l,ls, F, D, dL, S, h, N, l_Moon, b_Moon;
  Vec3D  e_Moon;
  // Средние значения элементов лунной орбиты
  L_0 = Frac (0.606433 + 1336.855225*T);       // Средняя долгота в полных оборотах
  l   = pi2*Frac ( 0.374897 + 1325.552410*T ); // Средняя аномалия Луны
  ls  = pi2*Frac ( 0.993133 +   99.997361*T ); // Средняя аномалия Солнца
  D   = pi2*Frac ( 0.827361 + 1236 853086*T ); // Разница долгот Луна-Солнце
  F   = pi2*Frac ( 0 259086 + 1342.227825*T ); // Расстояние от восходящего узла
  // Возмущения в долготе и широте
  dL = +22640*sin(l) - 4586*sin(l-2*D) + 2370*sin(2*D) +  769*sin(2*l)
       -668*sin(ls) - 412*sin(2*F) - 212*sin(2*l-2*D) - 206*sin(l+ls-2*D)
       +192*sin(l+2*D) - 165*sin(ls-2*D) - 125*sin(D) - 110*sin(l+ls)
```

```
      +148*sin(l-ls) - 55*sin(2*F-2*D),
  S  = F + (dL+412*sin(2*F)+541*sin(ls)) / Arcs;
  h  = F-2*D;
  N  = -526*sin(h) + 44*sin(l+h) - 31*sin(-l+h) - 23*sin(ls+h)
      + 11*sin(-ls+h) - 25*sin(-2*l+F) + 21*sin(-l+F);

  // Эклиптические долгота и широта
  l_Moon = pi2 * Frac( L_0 + dL/1296.0e3 ); // [rad]
  b_Moon = ( 18520.0*sin(S) + N ) / Arcs;   // [rad]
  // Экваториальные координаты
  e_Moon = R_x(-eps) * Vec3D(Polar(l_Moon,b_Moon));
  RA  = e_Moon[phi];
  Dec = e_Moon[theta];
}
//------------------------------------------------------------------------
// MiniSun:  Вычисляет экваториальные координаты Солнца,
//           используя ограниченные по точности ряды
//   T         Время в юлианских столетиях от эпохи J2000
//   RA        Прямое восхождение Солнца [рад]
//   Dec       Склонение Солнца [рад]
//------------------------------------------------------------------------
void MiniSun (double T, double& RA, double& Dec)
{
  const double eps = 23.43929111*Rad;
  double L,M;
  Vec3D  e_Sun;
// Средняя аномалия и эклиптическая долгота
  M  = pi2 * Frac ( 0.993133 + 99.997361*T);
  L  = pi2 * Frac ( 0.7859453 + M/pi2 +
                    (6893.0*sin(M)+72.0*sin(2.0*M)+6191.2*T) / 1296.0e3);
  // Экваториальные координаты
  e_Sun = R_x(-eps) * Vec3D(Polar(L,0.0));
  RA  = e_Sun[phi];
  Dec = e_Sun[theta];
}
```

3.3. Звездное время и часовой угол

Для вычисления высоты над горизонтом Солнца или Луны значение прямого восхождения непосредственно не используется. Однако оно нужно для определения часового угла светила, который равен разности прямых восхождений объекта и прямого восхождения звезды, пересекающей в данный момент небесный меридиан. Вращение Земли вокруг своей оси приводит к тому, что в течение суток через меридиан проходят объекты со всеми возможными значениями прямого восхождения. Каждый час прямое восхождение кульминирующих звезд увеличивается примерно на 1^h.

Повторяющееся изо дня в день движение звезд очень удобно для введения новой системы счета времени, называемой *звездным временем*. Для любой точки на Земле оно определяется как прямое восхождение звезд, в данный конкретный момент пересекающих местный небесный меридиан. Благодаря такому определению звездное время легко получить непосредственно из наблюдений.

Необходимость вводить звездное время наряду с солнечным возникает из-за небольшого различия в продолжительности солнечных суток и периода осевого вращения Земли. Привычные нам в повседневном обиходе часы делят сутки на 24 часа. Причем длительность суток соответствует периодичности чередования света и темноты, которая определяется движением Солнца. 24 часа соответствуют среднему времени между двумя последовательными прохождениями Солнца через небесный меридиан. Однако если мы измерим аналогичный интервал времени для звезд, то он окажется равным $23^h56^m4\overset{s}{.}091$. Этот немного более короткий период в отличие от солнечных суток называется звездными сутками, и он в точности равен длительности одного оборота Земли вокруг своей оси. Причина четырехминутного различия в длительности солнечных и звездных суток кроется в годичном орбитальном движении Земли вокруг Солнца. В результате этого движения прямое восхождение Солнца изменяется на $360° \equiv 24^h$ за год, что составляет примерно 4^m в сутки. Вследствие этого время между двумя последовательными кульминациями Солнца оказывается больше, чем аналогичное время для звезд.

Для любого момента Всемирного времени (UT) звездное время в Гринвиче можно вычислить по формуле:

$$\begin{aligned}\Theta_0 &= 24110\overset{s}{.}54841 + 8640184\overset{s}{.}812866 \cdot T_0 + 1{,}0027379093 \cdot UT + \\ &+ 0\overset{s}{.}093104 \cdot T^2 - 0\overset{s}{.}0000062 \cdot T^3,\end{aligned} \qquad (3.6)$$

где

$$T_0 = \frac{\mathrm{JD}_0 - 2451545}{36525} \quad \text{и} \quad T = \frac{\mathrm{JD}_0 - 2451545}{36525}.$$

JD и JD_0 обозначают соответственно юлианскую дату момента наблюдения и юлианскую дату на 0^h UT в день наблюдения.

```
//------------------------------------------------------------------------
// GMST: Среднее гринвичское звездное время
//   MJD        Время в форме модифицированной юлианской даты
// <return>:   GMST в радианах
//------------------------------------------------------------------------
double GMST (double MJD)
{
  const double Secs = 86400.0;         // Число секунд в сутках
  double MJD_0, UT, T_0, T, gmst;
  MJD_0 = floor ( MJD );
  UT    = Secs*(MJD-MJD_0);        // [сек]
  T_0   = (MJD_0-51544.5)/36525.0;
  T     = (MJD  -51544.5)/36525.0;
  gmst  = 24110.54841 + 8640184.812866*T_0 + 1.0027379093*UT
          + (0.093104-6.2e-6*T)*T*T;      // [сек]
  return  (pi2/Secs)*Modulo(gmst,Secs);   // [рад]
}
```

Для места с географической долготой λ местное звездное время будет отличаться от гринвичского звездного на $\lambda/15°$ часов:

$$\Theta = \Theta_0 + \lambda \cdot 1^h / 15°. \qquad (3.7)$$

Здесь долгота λ считается *положительной к востоку* от Гринвича (например, долгота Мюнхена $\lambda = +11^\circ\!.6$). Часовой угол τ звезды можно определить по ее прямому восхождению по формуле:

$$\tau = \Theta - \alpha. \tag{3.8}$$

В качестве примера того, как пользоваться звездным временем для перевода прямого восхождения в часовой угол, рассмотрим фрагмент программы, вычисляющий азимут и высоту Денеба (α Лебедя) для заданного места и времени наблюдения:

```
double lambda = +11.6*Rad;                  // Место наблюдения - Мюнхен
double phi    = +48.1*Rad;
double RA     = Ddd(20,41,12.8)<-Rad        // Экваториальные координаты Денеба
double Dec    = Ddd(45,15,25.8)*Rad
double ModJD  = Mjd ( 1993,8,1, 21,0,0.0 ); // 1 августа 1993, 21:00 UT

double tau,h,Az;

tau = GMST(ModJD) + lambda - RA;            // Часовой угол [рад]
Equ2Hor ( Dec,tau,phi, h,Az );              // Азимут и высота
cout << "Deneb:" << endl                    // Вывод
     << setprecision(2)
     << "Altitude   =" << setw(7) << Deg*h << " deg"
     << "Azimuth    =" << setw(7) << Deg*Az « " deg";
```

3.4. Всемирное и эфемеридное время

Мы уже довольно долго пользуемся понятием «время», однако до сих пор не уделили внимания некоторым важным вопросам. В астрономии приходится сталкиваться с использованием целого ряда различных систем счета времени. По наиболее общим признакам их можно разделить на два класса, наиболее важными представителями которых являются *динамическое время* (TT, TDB) и Всемирное время (UT). Чтобы разобраться, чем они отличаются друг от друга и каковы причины их использования, нам понадобится некоторое терпение.

В основу представления о динамическом времени положено требование, чтобы все астрономические процессы корректно описывались с точки зрения физики. Одна секунда динамического времени определяется через частоту электромагнитного излучения, испускаемого при определенном переходе в атоме цезия, и измеряется при помощи атомных часов. С 1984 года динамическое время заменило использовавшееся ранее эфемеридное время (ET), которое определялось путем сравнения наблюдаемых положений тел Солнечной системы с предвычисленными по законам классической механики координатами. Основанием для перехода к динамическому времени послужили релятивистские эффекты. Теория относительности установила, что не существует универсального абсолютного времени и что скорость хода часов зависит от положения и движения системы отсчета, относительно которой выполняется измерение времени. Отсюда, в частности, возникает необходимость различать земное (динамическое) время (TT или

TDT), отнесенное к центру Земли, и динамическое время барицентра (TDB), связанное с центром масс Солнечной системы. К счастью, для наших целей можно пренебречь различием между этими тремя временами — ET, TT и TDB.

В отличие от эфемеридного и динамического времени, Всемирное время (UT) пользуется неоднородной шкалой. UT — это наиболее совершенный вариант солнечного времени. Когда его вводили, то исходили из предположения, что средняя длительность суток остается неизменной на протяжении по крайней мере нескольких тысячелетий и составляет 24 часа. В результате получилось, что длительность секунды Всемирного времени не остается постоянной, поскольку реальная средняя продолжительность суток зависит от вращения Земли и видимого движения Солнца (то есть от длительности года). К сожалению, не существует универсального способа определения Всемирного времени по динамическому, поскольку вариации вращения Земли не поддаются точному прогнозированию. Каждое изменение скорости вращения Земли меняет продолжительность суток и требует учета при определении UT. Всемирное время поэтому определяется как функция звездного времени, которое непосредственно отражает неравномерность вращения Земли. Для любых суток 0^h UT определяется как момент, когда гринвичское среднее звездное время (GMST) имеет значение

$$GMST(0^h UT) = 24110{,}^s54841 + 8640184{,}^s812866 \cdot T_0 + \\ + 0{,}^s093104 \cdot T_0^2 - 0{,}^s0000062 \cdot T_0^3,$$

где

$$T_0 = \frac{\mathrm{JD}(0^h UT) - 2451545}{36525}.$$

Последнюю формулу мы уже встречали в разделе 3.3, где она использовалась для вычисления звездного времени по заданному моменту Всемирного. Теперь, однако, мы знаем, что только звездное время можно непосредственно получить из наблюдений, тогда как Всемирное время определяется на его основе.

Разность между Всемирным временем и эфемеридным, или динамическим, можно узнать только ретроспективно, то есть для моментов в прошлом. В таблице 3.1 дана сводка изменения ее величины $\Delta T = \mathrm{ET} - \mathrm{UT}$ (ET = TT = TDB) на протяжении XX века. В настоящее время ΔT увеличивается на 0,5–1,0 с в год.

Таблица 3.1. Значение ΔT = ET – UT (в секундах) в период с 1900 до 2000 гг.

Год	ET-UT	Год	ET-UT	Год	ET-UT	Год	ET-UT
1900	−2,72						
1905	3,86	1930	24,02	1955	31,07	1980	50,54
1910	10,46	1935	23,93	1960	33,15	1985	54,34
1915	17,20	1940	24,33	1965	35,73	1990	56,86
1920	21,16	1945	26,77	1970	40,18	1995	60,82
1925	23,62	1950	29,15	1975	45,48	2000	(65,0)

Время, которым мы пользуемся в повседневной жизни, основывается на координатном всемирном времени (UTC). UTC отсчитывается атомными часами

в том же темпе, что и земное динамическое время. Однако для согласования его со Всемирным временем в его отсчет один или два раза в год добавляют вставную секунду. Этим обеспечивается то, что UTC никогда не отклоняется от UT более чем на 0,9 секунды. На всей Земле именно UTC является основой для измерения времени. Поверхность Земли разделена на часовые пояса, в которых гражданское время отличается от UTC на целое (или полуцелое) число часов. Границы часовых поясов проведены с учетом географических особенностей территорий, в первую очередь государственных границ, чтобы по возможности избежать попадания одной страны в несколько часовых поясов. Границы часовых поясов закреплены международным соглашением, а поясное время для любой территории можно определить по карте часовых поясов.

Теперь самое время спросить, а каким же временем мы должны пользоваться при определении моментов восхода и захода Солнца и Луны? На основе приведенных определений можно сформулировать следующие правила применения Всемирного и эфемеридного времени:

- Всемирное время (UT) используется для вычисления звездного времени.
- Эфемеридное время (ET), или динамическое время (TT, TDB), используется для вычисления эфемерид Солнца, Луны и планет.

В качестве примера рассмотрим задачу вычисления высоты Луны и последовательно все шаги расчета. Допустим, нас интересует момент 0^h 1 января 1982 года по центрально-европейскому времени, а в качестве места наблюдения выберем Мюнхен ($\varphi = 48{,}^\circ1$, $\lambda = -11{,}^\circ6$). Мы можем игнорировать незначительную разницу между UTC и UT и считать, что Всемирное время UT = CET – 1^h. Как сказано выше, Всемирное время используется для вычисления звездного времени, в то время как для определения координат Луны надо использовать эфемеридное время. Это учитывается в приводимом ниже фрагменте программы.

```
double lambda = +11.6*Rad;                        // Географические координаты Мюнхена
double phi    = +48.1*Rad;
double MjdCET = Mjd ( 1982,1,1, 0,0,0.0 );        // 1 января 1982, 00 00 CET
double CET_UT =1.0                                // Поясное время (в часах)
ET_UT =52 17;                                     // в секундах на начало 1982
double MjdUT,MjdET, RA,Dec, sinH;
MjdUT = MjdCET - CET_UT/24.0;                     // MJD по Всемирному времени
MjdET = MjdUT + ET_UT/86400.0;                    // MJD по эфемеридному времени
MiniMoon ( MjdET,RA,Dec);                         // Экваториальные координаты
tau = GMST(MjdUT) + lambda - RA;                  // Часовой угол [рад]
sinH = sin(phi)*sin(Dec)                          // Синус высоты
     + cos(phi)*cos(Dec)*cos(tau);                //    над горизонтом
```

Может возникнуть вопрос, насколько большие ошибки появятся, если игнорировать различие между Всемирным и эфемеридным временем. Если использовать такое упрощение, то приведенный только что пример станет выглядеть так:

```
MjdUT = MjdCET - CET_UT/24.0;                     // MJD по Всемирному времени
MiniMoon (MjdUT,HA,Dec);                          // Экваториальные координаты
tau = GMST(MjdUT) + lambda - RA;                  // Часовой угол [рад]
sinH = sin(phi)*sin(Dec)                          // Синус высоты
     + cos(phi)*cos(Dec)*cos(tau);                //    над горизонтом
```

Хотя звездное время по-прежнему вычислено правильно, положение Луны определено на момент, который на ΔT = ET – UT *раньше* требуемого. В настоящее время ΔT имеет величину около одной минуты. За это время Луна перемещается примерно на 30 угловых секунд. Эта ошибка даже меньше той, которая возникает из-за использования упрощенной функции MiniMoon. Возможная ошибка при определении моментов восхода и захода составляет около 3 секунд. Для Солнца эта ошибка будет еще меньше. Поэтому в программе Sunset мы можем с чистой совестью игнорировать ΔT. Однако в других задачах, например при точном вычислении положений планет или для предвычисления покрытий звезд, необходимо с большим вниманием относиться к различию систем счета времени. Вот почему мы уделили здесь так много внимания этому вопросу.

3.5. Параллакс и рефракция

До сих пор мы пользовались геоцентрическими экваториальными координатами, то есть системой координат, начало которой находится в центре Земли. Однако свои наблюдения вы ведем не из центра Земли, а с ее поверхности. Какие поправки нужно внести в вычисленные нами значения координат, чтобы определить видимые положения наблюдаемых объектов?

При наблюдении удаленных звезд можно смело пренебречь различием между геоцентрическими и *топоцентрическими* координатами, которые получают путем параллельного переноса системы координат из центра Земли в выбранную точку ее поверхности. По сравнению с размерами Земли расстояние до звезд можно считать бесконечно большим. Однако, имея дело с объектами Солнечной системы, это различие, называемое *параллаксом*, следует учитывать.

Параллакс проявляется в том, что объект при наблюдении с поверхности виден несколько ниже над горизонтом, чем был бы виден при наблюдении из центра Земли. То есть топоцентрическая высота h немного меньше геоцентрической высоты h' (рис. 3.2). В зените параллакс объекта обращается в ноль. Однако с уменьшением высоты параллакс увеличивается, достигая максимума на высоте $h = 0°$.

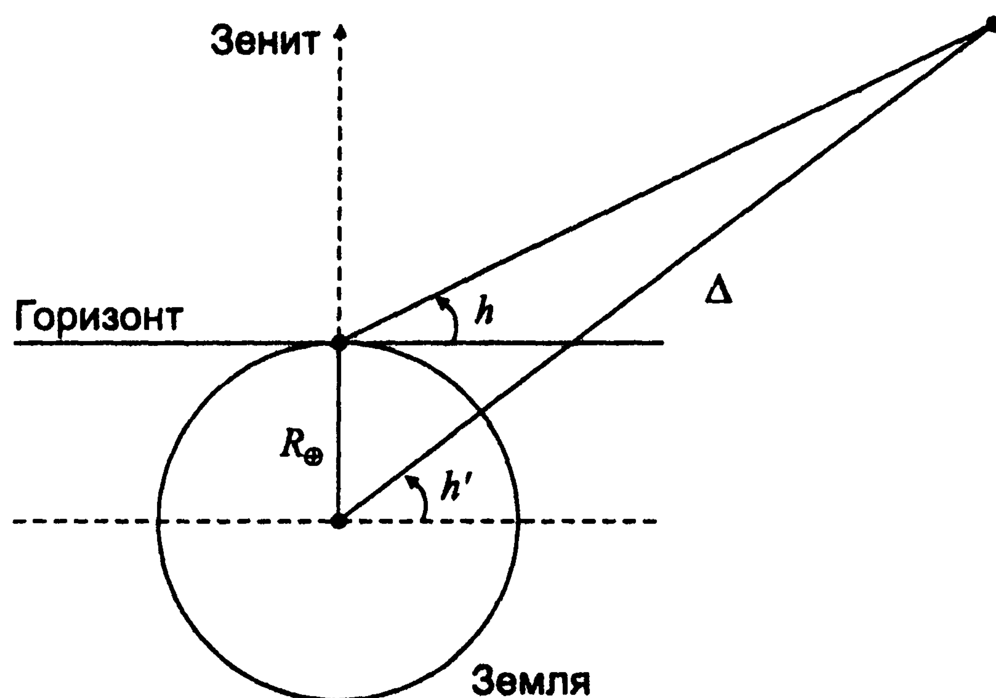


Рис. 3.2. Параллакс

Для вычисления горизонтального параллакса можно пользоваться следующей формулой

$$\pi = \arcsin(R_{\oplus}/\Delta), \tag{3.9}$$

где $R_{\oplus} \approx 6378$ км — расстояние наблюдателя от центра Земли, а Δ — *геоцентрическое* расстояние до объекта. Если подставить в эту формулу расстояние до Солнца, то получим значение параллакса $\pi_{\odot} = 8''\!.8$. В то же время для Луны — ближайшего нашего соседа — суточный параллакс достигает внушительной величины

$$\pi_{☾} = \arcsin\left(\frac{6378\,\text{км}}{384400\,\text{км}}\right) \approx 57'.$$

Понятно, что для определения моментов восхода и захода мы должны использовать не геоцентрическую высоту, а топоцентрическую. Но прежде чем переходить к программированию, рассмотрим еще один важный эффект.

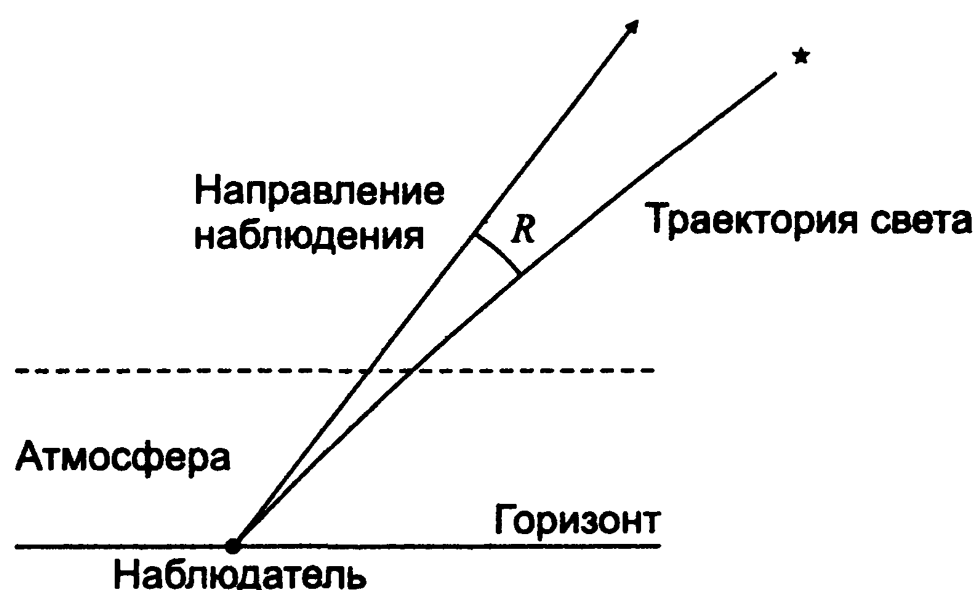


Рис. 3.3. Рефракция

Попадая из космического вакуума в плотные слои атмосферы, лучи света испытывают преломление и искривляются в вертикальной плоскости (рис. 3.3). Это явление называется атмосферной *рефракцией*. Оно приводит к тому, что для наблюдателя на поверхности Земли звезды немного «приподнимаются» над горизонтом. Лучи света, входящие в атмосферу под малым углом, проходят более длинный путь в слоях воздуха с различной плотностью (а значит, и показателями преломления) по сравнению с лучами, идущими вертикально. Поэтому атмосферная рефракция больше для объектов, видимых на малой высоте. Максимальное значение — около 34′ — достигается для объектов, находящихся на горизонте. Данные для других высот приведены в табл. 3.2. Для определения видимой высоты объекта к вычисленной геоцентрической высоте надо добавить поправку за рефракцию R. Величина атмосферной рефракции зависит от плотности и температуры воздуха. Детальные таблицы нормальной рефракции и формулы для ее приближенного вычисления нетрудно найти в литературе.

Таблица 3.2. Значения рефракции вблизи горизонта

h	10°	5°	2°	1°	0°
R	5′31″	10′15″	19′7″	25′36″	34′

Итак, мы видим, что для определения моментов восхода и захода нам следует использовать топоцентрические координаты, а также учитывать поправку за рефракцию в горизонте. Кроме того, поскольку моменты восхода и захода всегда определяются по верхнему краю лимба Солнца или Луны, нам также следует учесть соответствующий видимый радиус $s_{\odot}$ или $s_{☾}$. В результате мы получаем, что в моменты восхода и захода объект имеет геоцентрическую высоту

$$h_{R/S} = 0° + \pi - R_{h=0} - s. \tag{3.10}$$

Таким образом, наша задача состоит теперь в определении моментов времени, когда в заданный день наблюдаемый объект будет находиться на высоте $h = h_{R/S}$. Поскольку в нашем случае точность выше нескольких минут дуги не имеет смысла, мы можем не вычислять π и s точно, а пользоваться средними значениями. При вычислении восходов и заходов обычно используют следующие величины:

- для Солнца: $h_{R/S} = -0°50'$,
- для Луны: $h_{R/S} = +0°08'$,
- для звезд и планет: $h_{R/S} = -0°34'$.

Наша программа также вычисляет моменты начала и конца сумерек, которые определяются следующим образом:

- астрономические сумерки $h_{\odot} = -18°$,
- навигационные сумерки $h_{\odot} = -12°$,
- гражданские сумерки $h_{\odot} = -6°$.

Здесь никаких поправок за параллакс, рефракцию или радиус диска вводить не требуется. Сумерки по определению начинаются и заканчиваются в моменты, когда *геоцентрическая* высота Солнца достигает приведенных значений. Итак, нам остается теперь решить общую задачу по определению моментов времени, когда небесное тело имеет определенную высоту над горизонтом.

3.6. Моменты восхода и захода

Уравнение (3.5) позволяет определить высоту по заданным географической широте φ, склонению δ и часовому углу τ. Часовой угол для заданного момента времени можно получить по формуле (3.8).

Формулу (3.5) нетрудно преобразовать так, чтобы по известной высоте определять часовой угол:

$$\cos\tau = \frac{\sin h - \sin\varphi \sin\delta}{\cos\varphi \cos\delta}. \tag{3.11}$$

Пользуясь этой формулой, мы можем вычислить часовой угол звезды, когда она видна на высоте h над горизонтом. Надо учитывать, что данное уравнение имеет смысл не для всех значений h, а только для тех, которых объект реально может достигнуть. Например, звезда со склонением $\delta = 60°$ на географической широте $\varphi = 50°$ является незаходящей, а ее высота меняется в интервале от $h = 20°$ до

$h = 80°$. Если для некоторого значения h вычисленная по формуле (3.11) величина получается больше 1, это означает, что данная звезда постоянно остается либо выше, либо ниже данной высоты.

Для вычисления моментов восхода и захода звезд подставим в уравнение (3.11) значение высоты $h_{R/S} = -0°34'$, а полученное из него значение часового угла выразим в единицах времени ($15° \equiv 1^h$). Именно такой интервал звездного времени, потребуется восходящей звезде для того, чтобы достичь кульминации. Звездные сутки соответствуют $23^h56^m4^s\!,091$ солнечного времени, что дает коэффициент $23^h56^m4^s\!,091/24^h = 0{,}9972696$ для перевода интервалов звездного времени в интервалы солнечного. Если умножить на него величину часового угла, то мы получим значение, равное половине того времени, которое звезда видна над горизонтом. Для звезды с прямым восхождением $\alpha_\star$ звездное время в моменты восхода и захода равно

$$\Theta_{R/S} = \begin{cases} a_\star - \tau & \text{для восхода,} \\ a_\star + \tau & \text{для захода.} \end{cases}$$

Обозначив Θ_0 местное звездное время в полночь, получим, что звезда восходит за $0{,}9973 \cdot (\Theta_0 - \Theta_R)$ до полуночи и заходит через $0{,}9973 \cdot (\Theta_S - \Theta_0)$ после полуночи. В отличие от звезд, координаты которых остаются постоянными, положения Солнца и Луны значительно изменяются в течение дня. В то же время, для того чтобы определить по приведенным уравнениям моменты восхода и захода, нам нужно знать прямое восхождение и склонение объекта на момент пересечения им линии горизонта. Здесь приходится использовать итеративную процедуру. Сначала берутся координаты объекта на произвольный момент нужного нам дня (обычно используют $t = 12^h$) и вычисляются приближенные значения моментов восхода и захода. Для этих моментов вновь вычисляют координаты объекта и по ним определяют уточненные моменты восхода и захода. Для Луны эту процедуру повторяют до тех пор, пока разница значений, полученных на последовательных шагах, не станет меньше минуты. Для Солнца и планет уже после первой итерации получаются достаточно точные результаты.

К сожалению, на этом трудности еще не заканчиваются. При определении моментов восхода и захода Луны встречаются ситуации, в которых простой итеративный метод сталкивается с определенными затруднениями. Пользуясь им, нужно иметь в виду следующее.

- Поскольку Луна вращается вокруг Земли с запада на восток, она остается на небе дольше, чем звезды с тем же, что и у Луны, склонением. В результате восход Луны каждый день запаздывает примерно на 50^m по сравнению с предыдущими сутками. Из этого следует, что в каждом месяце есть такой день, когда Луна не восходит, а также другой день, когда она не заходит. Например, 8 февраля 1988 года в Мюнхене Луна взошла в 23^h30^m, а зашла в 9^h41^m следующего дня. Вследствие своего движения относительно звезд она затем оставалась под горизонтом позже полуночи и взошла только в 0^h43^m 10 февраля. 9 февраля восхода Луны не было.

- В высоких географических широтах Солнце и Луна часто остаются над горизонтом много дней. Моменты их восхода и захода определяются, главным образом, постепенным изменением склонения. В таких случаях определить долгожданные моменты касания горизонта при помощи итерационного метода становится трудно.

По этим причинам в программе Sunset используется другой принцип расчета. Моменты восхода и захода определяются путем обратной интерполяции таблиц высоты Солнца и Луны. Это приводит к большему объему вычислений, но позволяет значительно упростить структуру программы.

3.7. Квадратичная интерполяция

Если мы построим таблицу высот с шагом в один час, то, используя простую интерполяционную формулу, мы сможем получать по ней значения высоты как функции времени. Для наших целей подходит квадратичная интерполяция: по трем значениям мы можем вычислить коэффициенты параболы, аппроксимирующей функцию между выбранными точками. Обозначим табулированные значения функции

$$y_- = f(x = -1), \quad y_0 = f(x = 0) \quad \text{и} \quad y_+ = f(x = +1).$$

Нам нужно определить коэффициенты a, b и c параболы

$$y = a \cdot x^2 + b \cdot x + c, \tag{3.12}$$

которая проходит через точки $(-1, y_-)$, $(0, y_0)$ и $(1, y_+)$. Подставляя эти значения в формулу параболы, получаем три уравнения:

$$\begin{aligned} y_- &= a - b + c, \\ y_0 &= c, \\ y_+ &= a + b + c, \end{aligned}$$

из которых определяем искомые значения коэффициентов параболы

$$\begin{aligned} a &= (y_+ + y_-)/2 - y_0, \\ b &= (y_+ - y_-)/2, \\ c &= y_0. \end{aligned} \tag{3.13}$$

В случае если $b^2 \geq 4ac$, эта парабола имеет два корня в точках

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.14}$$

и экстремум в точке

$$\begin{aligned} x_E &= -b/2a, \\ y_E &= a \cdot x_E^2 + b \cdot x_E + c. \end{aligned} \tag{3.15}$$

Функция Quad решает эти уравнения, а также определяет корни, попадающие в интервал от $x = -1$ до $x = +1$.

```
//------------------------------------------------------------------------
// Quad: Квадратичная интерполяция
//       Находит корни и экстремум параболы,
//       интерполирующей три эквидистантных значения функции
//   y_minus   Значение функции в точке x = -1
//   y_0       Значение функции в точке x =  0
//   y_plus    Значение функции в точке x =  1
//   xe        Абсцисса экстремума (может лежать за границами [-1, 1])
//   ye        Значение функции в точке xe
//   root1     Первый найденный корень
//   root2     Второй найденный корень
//   n_root    Количество корней в интервале [-1, 1]
//------------------------------------------------------------------------
void Quad ( double y_minus, double y_0, double y_plus, double& xe, double& ye,
            double& root1, double& root2, int& n_root )
{
  double a,b,c, dis, dx;
  n_root = 0;
  // Коэффициенты интерполирующей параболы y=a*x^2+b*x+c
  a  = 0.5*(y_plus+y_minus) - y_0;
  b  = 0.5*(y_plus-y_minus);
  c  = y_0;

  // Находим экстремум
  xe = -b/(2.0*a);
  ye = (a*xe+b) * xe + c;
  dis = b*b - 4.0*a*c; // Дискриминант уравнения y=a*x^2+b*x+c

  if (dis >= 0) // Парабола имеет корни
  {
    dx = 0.5 * sqrt (dis) / fabs (a);
    root1 = xe - dx;
    root2 = xe + dx;
    if (fabs(root1) <= 1.0) ++n_root;
    if (fabs(root2) <= 1.0) ++n_root;
    if (root1      < -1.0) root1 = root2;
  }
}
```

3.8. Программа Sunset

Программа Sunset вычисляет моменты восхода и захода Солнца и Луны, а также моменты начала и конца навигационных сумерек для периода длительностью 10 дней. Вводится начальная дата периода, географические координаты места наблюдения, а также разница между местным и Всемирным временем.

Поиск моментов различных событий выполняется методом, проиллюстрированным на рис. 3.4. Функция SinAlt вычисляет на каждый час синусы высоты Солнца и Луны. Функция Quad интерполирует эти значения и отыскивает корни. Как только найден корень, проверяется, где находились Солнце или Луна в начале суток — если под горизонтом, то имеет место восход, в противном случае — заход. Затем выполняется поиск следующего корня. Особым образом рассматривается ситуация, когда на одном интервале интерполяции обнаруживаются два

корня. Чтобы определить, какой из них соответствует восходу, а какой заходу, проверяется высота объекта в экстремуме параболы. Процесс выполняется, пока не будет достигнут конец суток. Такой подход не встречается с трудностями при работе с незаходящими объектами и с объектами, которые целый день остаются под горизонтом.

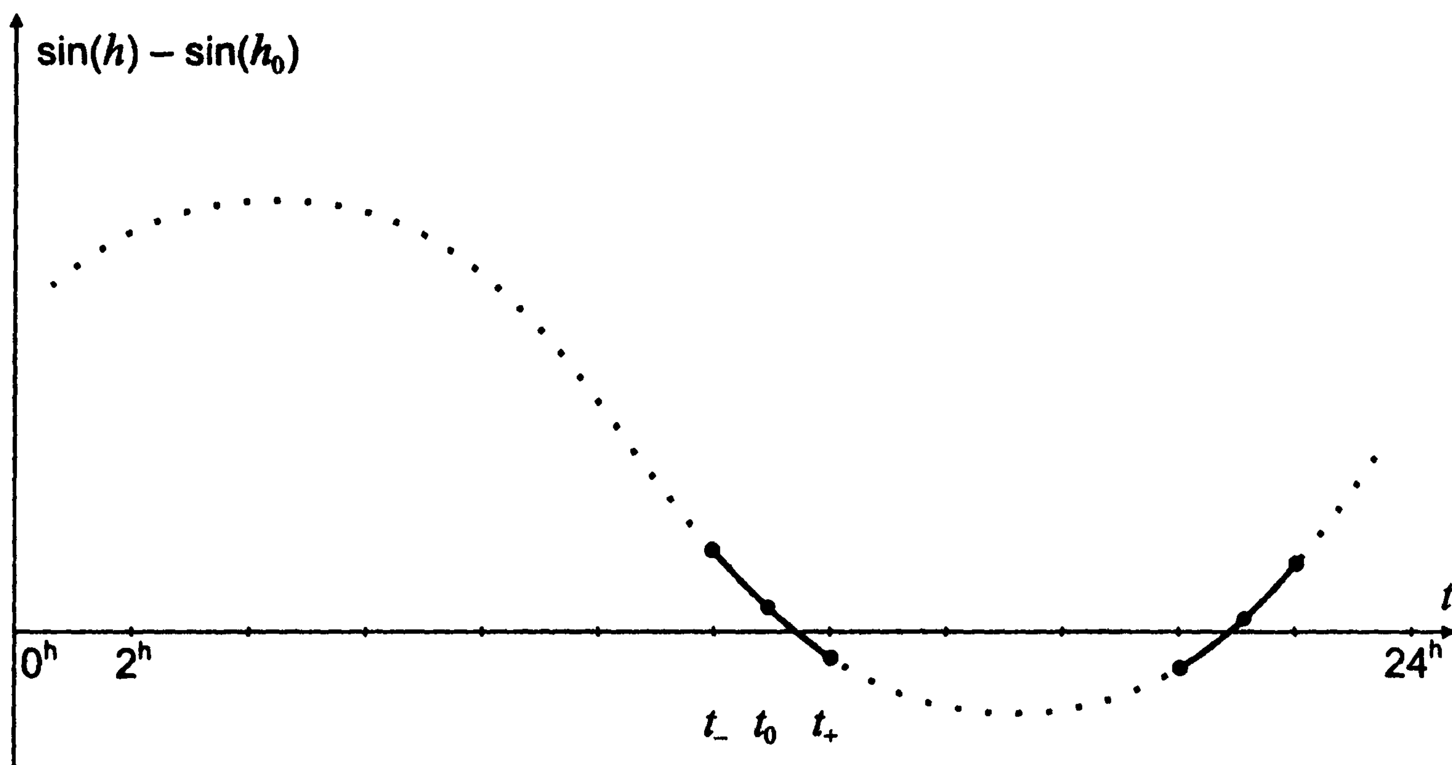


Рис. 3.4. Определение моментов восхода и захода

Аналогичным способом определяются моменты наступления гражданских, навигационных и астрономических сумерек с той лишь разницей, что из синуса высоты Солнца над горизонтом вычитаются соответственно константы $\sin(-6°)$, $\sin(-12°)$ или $\sin(-18°)$.

```
//---------------------------------------------------------------------------
// Файл:        Sunset.cpp
// Назначение: Моменты восхода и захода Солнца и Луны и наступления сумерек
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//---------------------------------------------------------------------------

#include <cmath>
#include <iomanip>
#include <iostream>

#include "APC_Const.h"
#include "APC_Math.h"
#include "APC_Moon.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"

#ifdef __GNUC__   // GNU C++ adaptation
#include <ctype.h>
#include "GNU_iomanip.h"
#endif

using namespace std;

// Типы искомых событий
enum enEvent {
```

```
  Moon,      // восходы и заходы Луны
  Sun,       // восходы и заходы Солнца
  CivilTwi,  // гражданские сумерки
  NautiTwi,  // навигационные сумерки
  AstroTwi   // астрономические сумерки
};

//------------------------------------------------------------------------------
// SinAlt: Синус высоты Солнца и Луны
// Input:
//   Event      Тип искомого события
//   MJD0       0h на расчетную дату (в форме модифицированной юлианской даты)
//   Hour       Час
//   lambda     Восточная географическая долгота (в радианах)
//   Cphi       Косинус географической широты
//   Sphi       Синус географической широты
// <return>:    Синус высоты Солнца или Луны в момент искомого события
//------------------------------------------------------------------------------
double SinAlt ( enEvent Event, double MJD0, double Hour,
                double lambda, double Cphi, double Sphi )
{
  double  MJD, T, RA, Dec, tau;
  MJD = MJD0 + Hour/24.0;
  T   = (MJD-51544.5)/36525.0;
  if ( Event == Moon )
    MiniMoon ( T, RA, Dec);
  else
    MiniSun ( T, RA, Dec );
  tau = GMST(MJD) + lambda - RA;
  return ( Sphi*sin(Dec)+Cphi*cos(Dec)*cos(tau) );
}

//------------------------------------------------------------------------------
// GetInput: Запрашивает ввод данных у пользователя
//   MJD       Время в форме модифицированной юлианской даты
//   lambda    Восточная географическая долгота наблюдателя (в радианах)
//   phi       Географическая широта наблюдателя (в радианах)
//   zone      Разница местного и Всемирного времени (в сутках)
//   twilight  Указывает на гражданские, навигационные или астрономические сумерки
//------------------------------------------------------------------------------
void GetInput ( double& MJD, double& lambda,
                double& phi, double& zone, enEvent& twilight )
{
  int  year, month, day;
  char cTwilight;
  cout << " First date (yyyy mm dd)                ";
  cin >> year >> month >> day; cin.ignore(81,'\n');
  cout << endl;
  cout << " Observing site:  east longitude [deg]   ";
  cin  >> lambda; cin.ignore(81,'\n');
  cout << "                  latitude [deg]       . ";
  cin  >> phi; cin.ignore(81,'\n');
  cout << "                  local time - UT [h]  . ";
  cin  >> zone; cin.ignore(81,'\n');
  cout << " Twilight definition (c, n or a)      . . ";
  cin  >> cTwilight; cin.ignore(81,'\n');
  switch (tolower(cTwilight)) {
    case 'c': twilight = CivilTwi; break;
```

```
    case 'a': twilight = AstroTwi; break;
    case 'n':
    default:  twilight = NautiTwi;
  }
  lambda*=Rad;
  phi*=Rad;
  zone/=24.0;
  MJD = Mjd(year,month,day) - zone;
}

//------------------------------------------------------------------------------
// FindEvents: Поиск моментов восхода/захода Солнца/Луны или начала/конца сумерек
//   Event     Указывает тип искомого события
//   MJD0h     0h на расчетную дату (в форме модифицированной юлианской даты)
//   lambda    Восточная географическая долгота наблюдателя (в радианах)
//   phi       Географическая широта наблюдателя (в радианах)
//   LT_Rise   Местное время восхода или начала сумерек
//   LT_Set    Местное время захода или конца сумерек
//   rises     Тип события
//   sets      Тип события
//   above     Солнце или Луна не заходят
//------------------------------------------------------------------------------
void FindEvents ( enEvent Event, double MJD0h, double lambda, double phi,
                  double& LT_Rise, double& LT_Set,
                  bool& rises, bool& sets, bool& above )
{
  static const double sinh0[5] = {
    sin(Rad*( +8.0/60.0)), // Восход Луны               h= +8'
    sin(Rad*(-50.0/60.0)), // Восход Солнца             h=-50'
    sin(Rad*(   - 6.0  )), // Гражданские сумерки       h=-6 deg
    sin(Rad*(   -12.0  )), // Навигационные сумерки     h=-12deg
    sin(Rad*(   -18.0  )), // Астрономические сумерки   h=-18deg
  };

  const double Cphi = cos(phi),
  const double Sphi = sin(phi);
  double hour = 1.0;
  double y_minus, y_0, y_plus;
  double xe, ye, root1, root2;
  int nRoot;

  // Инициализация поиска
  y_minus = SinAlt(Event, MJD0h, hour-1.0, lambda, Cphi, Sphi) - sinh0[Event];
  above = (y_minus>0.0);  rises = false;  sets  = false;

  // Перебор интервалов от [0h-2h] до [22h-24h]
  do {
    y_0    = SinAlt( Event, MJD0h, hour    , lambda, Cphi, Sphi )-sinh0[Event];
    y_plus = SinAlt( Event, MJD0h, hour+1.0, lambda, Cphi, Sphi )-sinh0[Event];
    // Определение параболы по трем значениям y_minus,y_0,y_plus
    Quad ( y_minus, y_0, y_plus, xe, ye, root1, root2, nRoot );
    if ( nRoot==1 ) {
      if ( y_minus < 0.0 )
        { LT_Rise = hour+root1;  rises = true; }
      else
        { LT_Set  = hour+root1;  sets  = true; }
    }
```

```
    if ( nRoot == 2 ) {
      if ( ye < 0.0 )
        { LT_Rise = hour+root2;  LT_Set = hour+root1; }
      else
        { LT_Rise = hour+root1;  LT_Set = hour+root2; }
      rises = true; sets = true;
    }
    y_minus = y_plus;      // подготовка к обработке следующего интервала
    hour += 2.0;
  }
  while ( !( ( hour == 25.0 ) || ( rises && sets ) ) );
}

//------------------------------------------------------------------------
//
// Основная программа
//
//------------------------------------------------------------------------
void main()
{
  // Переменные
  bool     above, rise, sett;
  int      iEvent, day;
  double   lambda, zone, phi;
  double   date, start_date, LT_Rise, LT_Set;
  enEvent  Event, Twilight;

  // Вывод заголовка
  cout << endl
       << "      SUNSET: solar and lunar rising and setting times  " << endl
       << "        (c) 1999 Oliver Montenbruck, Thomas Pfleger     " << endl
       << endl;
  // Ввод данных пользователем
  GetInput (start_date, lambda, phi, zone, Twilight );
  // Заголовок таблицы
  cout << endl
       << "    Date"
       << "             Moon              Sun              Twilight " << endl
       << "         "
       << "           rise/set          rise/set        beginning/end"
       << endl << endl;

  // Проход по десяти последовательным дням
  for (day=0; day<10; day++) {
    // Текущий день
    date = start_date + day;
    cout << " " << DateTime(date+zone) << "  ",

    // Перебор типов событий (Луна, Солнце, сумерки)
    for (iEvent=0; iEvent<=2; iEvent++)  {

      // После Солнца и Луны: задаем тип сумерек
      Event = (iEvent<2) ? (enEvent) iEvent : Twilight;

      // Теперь определяем моменты событий
      FindEvents ( Event, date, lambda, phi,
                   LT_Rise, LT_Set, rise, sett, above );
      // Вывод
```

```
      if ( rise || sett )  {
        if ( rise )
          cout << "   " << Time(LT_Rise.HHMM) << " ";
        else
          cout << "   ----- ";
        if ( sett )
          cout << "   " << Time(LT_Set.HHMM) << " ";
        else
          cout << "   ----- ";
      }
      else
        if ( above ) {
          if ( Event >= CivilTwi )
            cout << "    always bright ";
          else
            cout << "   always visible ";
        }
        else {
          if ( Event >= CivilTwi )
            cout << "     always dark  ";
          else
            cout << "  always invisible";
        }
    }
    cout << endl;
  }

  // Завершение
  cout << endl.
  cout << " all times in local time (=UT"
       << showpos << 24 0*zone << noshowpos << "h)"
       << endl;
}
```

Теперь рассмотрим использование программы Sunset на примерах, которые продемонстрируют некоторые тонкости процедуры вычисления моментов восхода и захода.

Программа Sunset вычисляет моменты восхода и захода на период длительностью 10 дней, начиная с указанной даты. Географическая широта считается положительной к востоку от гринвичского меридиана. Разница между местным и Всемирным временем задается в часах. Например, значение 1 соответствует центрально-европейскому времени (CET), 2 — центрально-европейскому летнему времени (CEST). Для часовых поясов в Северной Америке надо вводить отрицательные значения: например, −5 для восточного стандартного времени (EST) или −8 для тихоокеанского стандартного (PST). В завершение указывается тип сумерек, моменты начала и конца которых подлежат определению: 'c' задает гражданские сумерки, 'n' — навигационные, а 'a' — астрономические.

Допустим, нам нужно определить моменты восхода и захода Солнца и Луны в Мюнхене ($\lambda = 11{,}^\circ 6$ в. д., $\varphi = 48{,}^\circ 1$ с. ш.), начиная с 23 марта 2000 года. Мы запросим расчет по центрально-европейскому времени (разница со Всемирным временем 1^h). Запустив программу, вводим начальную дату в форме год, месяц, число, затем программа Sunset запрашивает географические координаты и разницу

между LT и UT (данные, вводимые пользователем, выделены курсивом). По введенным данным Sunset вычисляет моменты восхода и захода Солнца и Луны, а также моменты начала и конца навигационных сумерек (когда Солнце достигает высоты −12°).

```
     SUNSET: solar and lunar rising and setting times
       (c) 1999 Oliver Montenbruck, Thomas Pfleger

 First date (yyyy mm dd)              ... 2000 03 23

 Observing site:  east longitude [deg] ... +11.6
                  latitude [deg]       ... +48.1
                  local time - UT [h]  ... +1
 Twilight definition (c, n or a)       ... n
    Date            Moon            Sun             Twilight
                  rise/set        rise/set        beginning/end

 2000/03/23    22:12    08:01    06:10    18:31    05:02    19:39
 2000/03/24    23:17    08:28    06:08    18:32    05:00    19:41
 2000/03/25    -----    08:58    06:06    18:34    04:58    19:42
 2000/03/26    00:18    09:33    06:04    18:35    04:56    19:44
 2000/03/27    01:16    10:13    06:02    18:37    04:53    19:46
 2000/03/28    02:08    10:59    06:00    18:38    04:51    19:47
 2000/03/29    02:55    11:51    05:58    18:40    04:49    19:49
 2000/03/30    03:37    12:48    05:56    18:41    04:47    19:50
 2000/03/31    04:13    13:51    05:54    18:43    04:45    19:52
 2000/04/01    04:44    14:56    05:52    18:44    04:42    19:54

 all times in local time (=UT+1h)
```

Как мы видим, в графе Moon rise нет значения для 25 марта 2000 года. В эту ночь Луна восходит после полуночи, так что ее восход приходится уже на 26 марта. Обычно в каждом месяце бывает по одному дню, когда Луна не восходит или не заходит.

Чтобы проиллюстрировать другой интересный эффект, предположим, что нам нужно вычислить моменты восходов и заходов по центрально-европейскому летнему времени для точки наблюдения на широте 65°, начиная с 15 июня 1989 года. Sunset дает следующие результаты:

```
     SUNSET: solar and lunar rising and setting times
       (c) 1999 Oliver Montenbruck, Thomas Pfleger

 First date (yyyy mm dd)              ... 1989 06 15

 Observing site:  east longitude [deg] ... +10.0
                  latitude [deg]       ... +65.0
                  local time - UT [h]  ... +2.0
 Twilight definition (c, n or a)       ... n

    Date            Moon            Sun             Twilight
                  rise/set        rise/set        beginning/end

 1989/06/15    19:58    01:00    02:24    00:16    always bright
 1989/06/16    22:26    23:53    02:23    00:18    always bright
 1989/06/17   always invisible   02:22    00:19    always bright
```

```
1989/06/18   always invisible   02:21   00:20   always bright
1989/06/19   always invisible   02:20   00:21   always bright
1989/06/20   always invisible   02:20   00:22   always bright
1989/06/21     02:39    03:24   02:20   00:23   always bright
1989/06/22     01:35    06:21   02:20   00:23   always bright
1989/06/23     01:15    08:29   02:21   00:23   always bright
1989/06/24     01:01    10:25   02:22   00:22   always bright

all times in local time (=UT+2h)
```

Выясняется, что после захода 16 июня в 23^h53^m Луна остается ниже горизонта, то есть не восходит до 2^h39^m 21 июня. В высоких широтах Луна может многими днями оставаться под горизонтом.

Аналогичное поведение Солнца мы называем полярной ночью или полярным днем. Пользуясь программой Sunset, можно оценить продолжительность полярной ночи или полярного дня для любого места. Для этого подберем такой период, когда Солнце впервые становится always invisible (постоянно невидимо), и определим дату начала полярной ночи. Конец полярной ночи можно найти совершенно аналогичным способом.

Результат always bright (всегда светло) в колонке Twilight (сумерки) означает, что в течение всей ночи сумерки не прекращаются и полноценная темнота не наступает. Если мы будем определять моменты астрономических, а не навигационных или гражданских сумерек, то такое явление может иметь место не только в северных широтах, но даже на широте 50°.

3.9. Программа Planrise

В предыдущих разделах мы познакомились с инструментарием, используемым для вычисления восходов и заходов методом квадратичной интерполяции высот. Он удобен тем, что пригоден даже в тех особых случаях вычисления восходов и заходов, когда другие методы встречаются с определенными трудностями. Тем не менее мы все же хотим продемонстрировать программу, которая использует итеративный метод для вычисления моментов восходов и заходов планет. Снабженный комментариями исходный код программы Planrise содержится на прилагаемом к книге компакт-диске.

Работа программы Planrise основывается на уравнениях, приведенных в разделе 3.6. Для вычисления положений планет используется функция PertPosition, описанная в разделе 5.2. Чтобы определить моменты восхода и захода планеты мы сначала определяем (геоцентрические) экваториальные координаты α и δ для произвольного момента времени в течение дня. Затем из уравнения (3.11) определяем значение $\cos\tau(\varphi,\delta)$ для часового угла в момент восхода или захода. Если значение, полученное в правой части уравнения, по абсолютной величине больше единицы, это означает, что в данные сутки планета не восходит и не заходит. В таком случае Planrise определяет по склонению планеты и географической широте места наблюдения, находится ли планета постоянно над или под горизонтом.

Поскольку горизонтальный параллакс и видимый размер диска планеты практически не влияют на моменты восхода и захода планет, мы используем для их определения высоту $h_{R/S} = -0°34'$, которая учитывает только величину рефракции. Итеративный процесс уточнения моментов восхода и захода выполняется по формуле

$$t_{i+1} = t_i - 0{,}9973(\Theta(t_i) - \alpha(t_i) \pm \tau(\varphi, \delta(t_i))), \tag{3.16}$$

где

$$\tau = \arccos \frac{\sin h_{A/U} - \sin\varphi \sin\delta}{\cos\varphi \cos\delta}. \tag{3.17}$$

Здесь знак плюс берется при вычислении восходов, а минус — для заходов. $\Theta(t_i)$ и $\alpha(t_i)$ обозначают соответственно местное звездное время и прямое восхождение планеты для момента t_i на i-м шагу процедуры. Коэффициент 0,9973 используется для перевода интервалов звездного времени в интервалы Всемирного.

Как и при вычислении моментов восхода и захода, для определения момента кульминации можно воспользоваться уравнением:

$$t_{i+1} = t_i - 0{,}9973(\Theta(t_i) - \alpha(t_i)). \tag{3.18}$$

Чтобы сократить время вычислений, в программе `Planrise` сделаны различные упрощения. В частности, для получения координат планеты α и δ используется линейная интерполяция. Местное звездное время также определяется линейной интерполяцией значений на 0^h и 24^h выбранных суток.

В качестве примера рассмотрим вычисление моментов восхода и захода планет в Мюнхене ($\lambda = 11{,}^\circ 6$ в. д., $\varphi = 48{,}^\circ 1$ с. ш.) 31 декабря 1999 года. Все вводимые пользователем данные выделены курсивом.

```
PLANRISE: planetary and solar rising and setting times
          (c) 1999 Oliver Montenbruck, Thomas Pfleger

 Date (yyyy mm dd)                          ... 1999 12 31

 Observing site:  east longitude [deg] ... +11.6
                  latitude [deg]       ... +48.1
                  local time - UT [h]  ... +1.0

              rise    culmination    set

 Sun          08:04      12:16      16:29
 Mercury      07·33      11·37      15:41
 Venus        04.52      09·30      14:08
 Mars         10:33      15:35      20:37
 Jupiter      12:29      19:10      01:55
 Saturn       13:09      20:10      03:14
 Uranus       10:02      14:45      19:28
 Neptune      09:25      13:57      18:29
 Pluto        05:11      10:22      15:32

 all times in local time (=UT+1h)
```

Итак, при хорошей погоде невооруженным глазом можно увидеть три планеты: Марс, Юпитер и Сатурн.

4. Кометные орбиты

Положения больших планет обычно вычисляют и публикуют на годы вперед. При расчете движения комет, как правило, ограничиваются определением элементов орбиты и краткосрочными эфемеридами. Пять или шесть элементов орбиты описывают движение объекта в пространстве и времени гораздо компактнее, чем длинная таблица положений. Для вновь открытых комет элементы орбит обычно публикуются в циркуляре Международного астрономического союза (МАС). Кроме того, существуют каталоги, содержащие элементы орбит для известных периодических комет и малых планет.

Зная элементы орбиты, можно при помощи соответствующей программы вычислить положение кометы относительно Земли и Солнца на любой момент времени. Рассматриваемая в этой главе программа `Comet` не ограничивается эллиптическими орбитами, характерными для планет и астероидов. Она годится для работы с любыми кометными орбитами, среди которых встречаются параболические, околопараболические и гиперболические.

В следующих разделах мы рассмотрим движение комет относительно Солнца и плоскости эклиптики, пренебрегая возмущениями со стороны больших планет, поскольку учет этих возмущений потребовал бы слишком больших усилий. Различные преобразования координат, необходимые для последующего получения геоцентрических координат, были нами рассмотрены в главе 2.

4.1. Форма и ориентация кометной орбиты

Движение комет вокруг Солнца подчиняется тем же законам, что управляют орбитами больших планет.

1. Орбита представляет собой эллипс, параболу или гиперболу (то есть коническое сечение), в одном из фокусов которого находится Солнце. В частности, это означает, что орбита лежит в фиксированной плоскости.
2. Прямая, соединяющая Солнце с кометой (радиус-вектор), заметает равные площади за равные интервалы времени. Это утверждение известно как второй закон Кеплера (закон равных площадей).

Эти законы были открыты Иоганном Кеплером и в дальнейшем объяснены на основе закона всемирного тяготения Ньютона. Законы Кеплера в точности выполняются только в тех случаях, когда влиянием больших планет на движение кометы можно пренебречь по сравнению с солнечным притяжением.

Орбита лежит в одной плоскости по той причине, что солнечное притяжение всегда действует вдоль линии, соединяющей Солнце с кометой, то есть представ-

ляет собой центральную силу. За короткий интервал времени Δt комета, двигаясь со скоростью $\dot{\boldsymbol{r}}$, преодолеет расстояние $\dot{\boldsymbol{r}} \cdot \Delta t$ между точками $\boldsymbol{r}$ и $\boldsymbol{r} + \dot{\boldsymbol{r}} \cdot \Delta t$ (рис. 4.1). Вместе с Солнцем эти две точки определяют в пространстве плоскость. Поскольку сила солнечного притяжения действует только в этой плоскости, дальнейшая траектория движения кометы не может ее покинуть. Второй закон Кеплера также является прямым следствием центрального характера силы тяготения. Доказательство этого утверждения требует довольно сложных математических выкладок, и поэтому мы не станем его приводить.

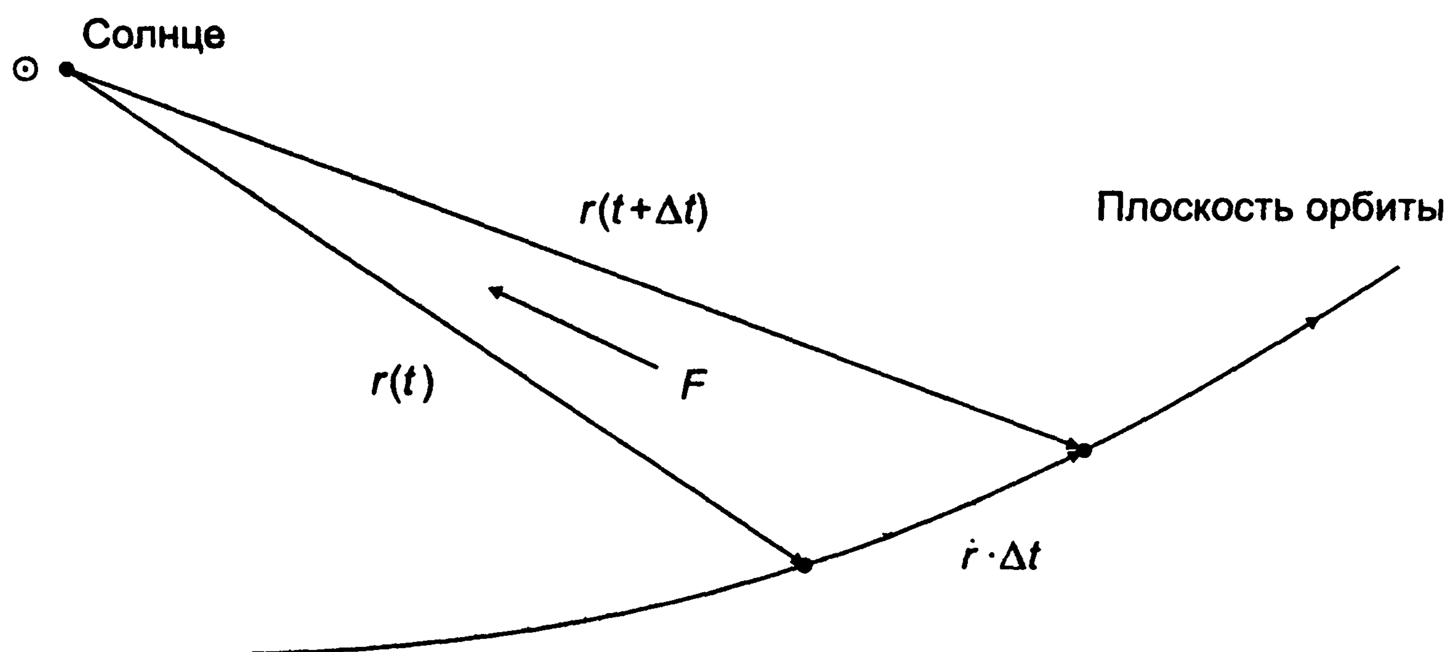


Рис. 4.1. Сила, действующая на комету, не меняет плоскость ее орбиты

Точная форма орбиты определяется ослаблением гравитационного притяжения по закону обратных квадратов ($F \sim 1/r^2$). Три основные формы — эллипс, парабола и гипербола — представлены на рис. 4.2. Кометы с эллиптическими орбитами являются периодическими и неоднократно возвращаются к Солнцу. Те же кометы, которые имеют параболические или гиперболические орбиты, лишь однажды на короткое время сближаются с Солнцем. Ближайшая к Солнцу точка орбиты называется *перигелием*. Прямая, соединяющая перигелий с Солнцем (*линия апсид*), делит орбиту на две симметричные части.

Точка на орбите определяется расстоянием от Солнца r и истинной аномалией[1] ν. Последняя представляет собой угол между линией, соединяющей Солнце с кометой, и направлением от Солнца к перигелию. Величины r и ν связаны между собой уравнением конического сечения:

$$r = \frac{p}{1 + e \cdot \cos\nu}. \tag{4.1}$$

Значение p, задающее расстояние при $\nu = 90°$, называется *орбитальным параметром* и определяет размер орбиты. *Эксцентриситет* e определяет, насколько орбита отклоняется от окружности ($e = 0$). Большее его значение соответствует более вытянутым орбитам. При $e = 1$ орбита становится параболической, а при $e > 1$ — гиперболической.

[1] В небесной механике термин «аномалия» используется в значении «угол». Помимо истинной аномалии используются средняя аномалия и эксцентрическая аномалия.

Для задания размера орбиты часто вместо орбитального параметра используется большая полуось, определенная в случае $e \neq 1$:

$$a = \frac{p}{1-e^2},\ p = a(1-e^2). \tag{4.2}$$

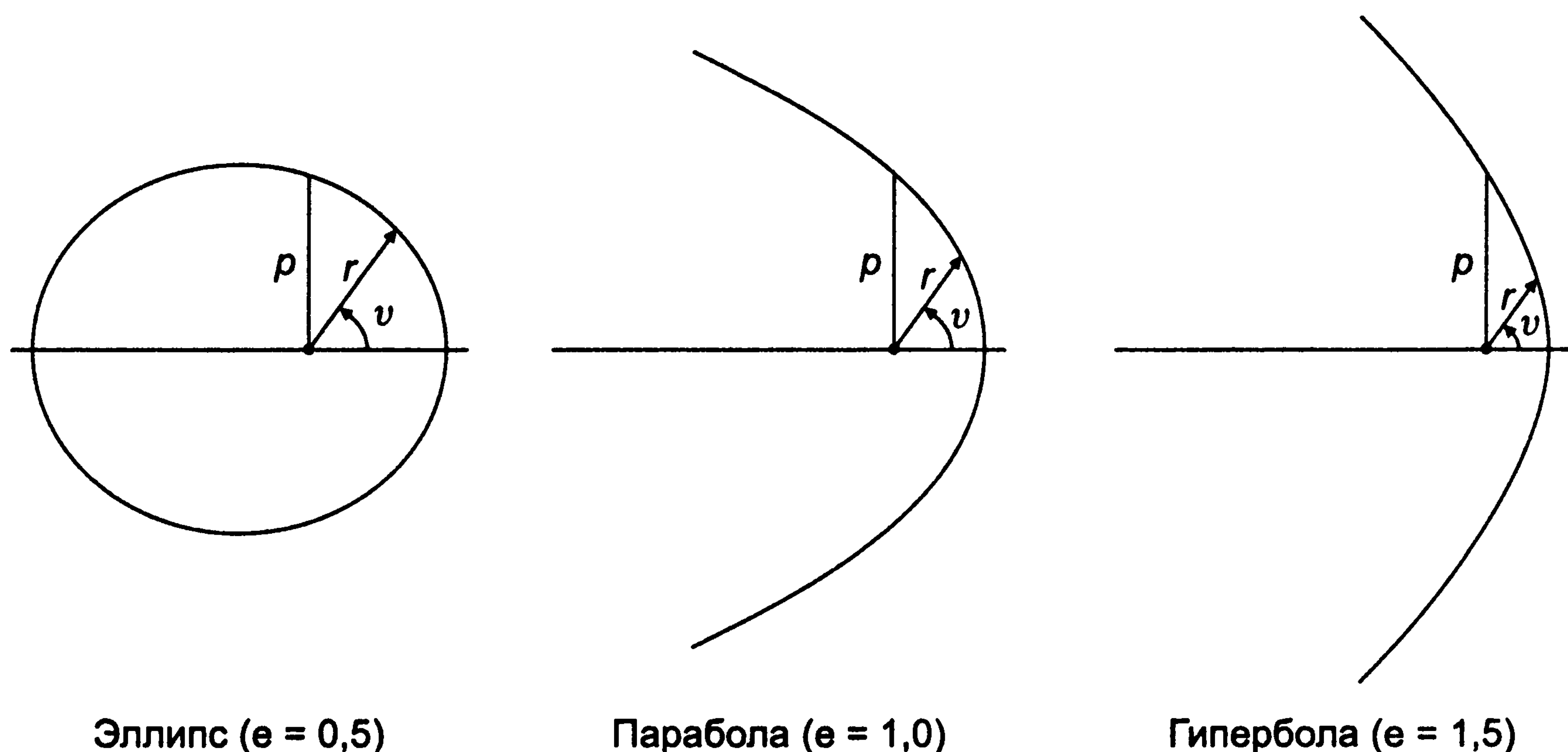


Рис. 4.2. Конические сечения с эксцентриситетами e = 0,5, e = 1,0 и e = 1,5 при одинаковом значении орбитального параметра p

Очевидно, что для эллиптических орбит большая полуось a равна среднему арифметическому между наименьшим и наибольшим расстояниями от Солнца:

$$r_{\min} + r_{\max} = \frac{p}{1+e} + \frac{p}{1-e} = \frac{2p}{1-e^2} = 2a.$$

Надо заметить, что большая полуось гиперболы (то есть при $e > 1$) по определению отрицательна. Для параболических орбит ($e \approx 1$) вместо a используется перигелийное расстояние $q = r_{\min}$:

$$q = \frac{p}{1+e}. \tag{4.3}$$

В случае строго параболической орбиты $q = p/2$.

4.2. Положение на орбите

Хотя уравнение конического сечения дает возможность определить все точки орбиты, через которые проходит комета, оно не позволит установить, в какой именно точке она будет находиться в заданный момент времени. Для решения этой задачи нужно использовать второй закон Кеплера. Рассмотрим основные этапы вычислений на примере замкнутой эллиптической орбиты.

Эллипс с эксцентриситетом e можно получить, уменьшив один из диаметров окружности так, чтобы он стал $\sqrt{1-e^2}$ от исходной величины. Эта же величина

определяет отношение площади эллипса с большой полуосью a к площади окружности радиусом a:

$$S = (\pi a^2)\sqrt{1-e^2}.$$

Линия, соединяющая Солнце с кометой (радиус-вектор), заметает всю эту площадь за интервал времени, равный периоду обращения. Отсюда получаем для сектора, заметенного с момента прохождения перигелия t_0 до текущего момента t

$$S(t) = (\pi a^2)\sqrt{1-e^2}\left(\frac{t-t_0}{T}\right). \qquad (4.4)$$

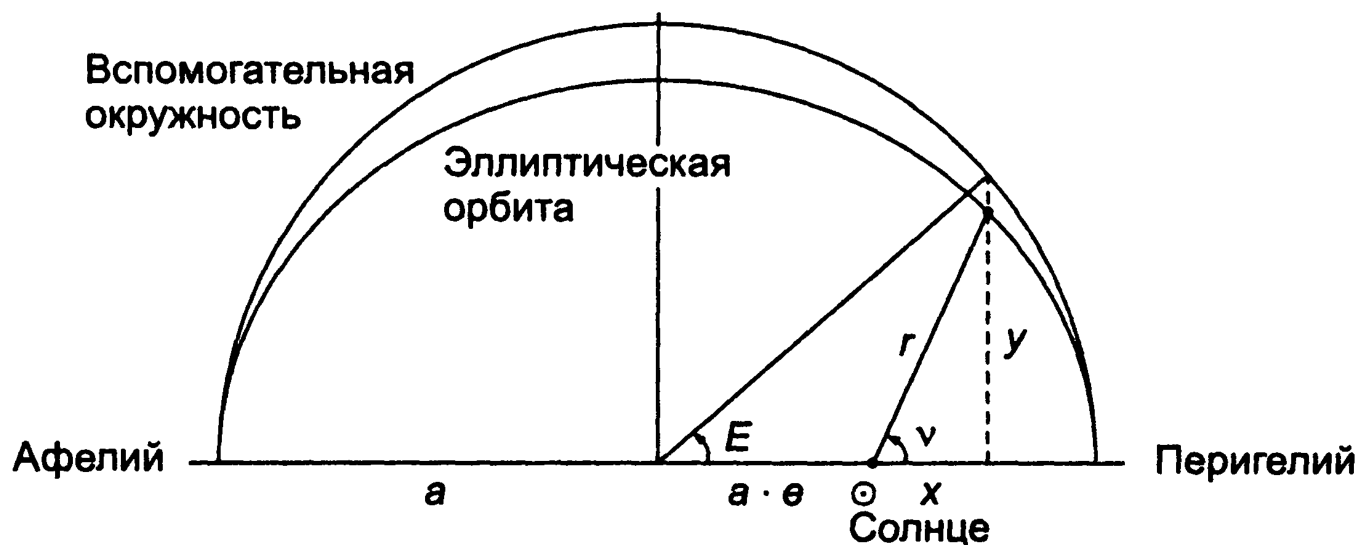


Рис. 4.3. Способ определения эксцентрической аномалии *E*

Если мы теперь сможем описать площадь этого сектора как функцию положения на орбите (r, ν), то получим соотношение, связывающее положение на орбите с моментом времени t. Для круговой орбиты все предельно просто: площадь S прямо пропорциональна углу ν и квадрату радиуса. Для решения этой задачи в случае эллипса введем вспомогательную величину — *эксцентрическую аномалию* E. Ее геометрический смысл показан на рис. 4.3. Если ось x выбрана, как изображено на рисунке, то абсцисса точки P, расположенной на эллипсе, будет равна $x = a \cdot \cos E - a \cdot e$. Здесь $a \cdot e$ соответствует расстоянию между центром эллипса и фокусом, в котором расположено Солнце. Координата y точки P отличается от ординаты точки P', расположенной на вспомогательной описанной окружности множителем $\sqrt{1-e^2}$, то есть $y = \sqrt{1-e^2} \cdot a \cdot \sin E$. Таким образом, мы получаем

$$\begin{aligned} x &= r\cos\nu = a(\cos E - e), \\ y &= r\sin\nu = a\sqrt{1-e^2}\sin E. \end{aligned} \qquad (4.5)$$

Для вычисления S рассмотрим две вспомогательные фигуры, показанные на рис. 4.4.

Сектор окружности ($P'O\Pi$): $S_1 = \pi a^2 E/360°$.

Сектор эллипса ($PO\Pi$): $S_2 = \sqrt{1-e^2} \cdot S_1$.

Треугольник ($O\odot P$): $S_3 = \frac{1}{2} y \cdot (a \cdot e) = \frac{1}{2} a^2 e\sqrt{1-e^2}\sin E$.

Сектор ($P\odot\Pi$): $S = S_2 - S_3$.

Пользуясь эксцентрической аномалией, площадь S можно выразить так:

$$S(t)=\frac{1}{2}a^2\sqrt{1-e^2}\left(E\frac{\pi}{180°}-e\sin E\right). \qquad (4.6)$$

Сравнивая это выражение для $S(t)$ с выведенным ранее (4.4), получаем соотношение между временем и эксцентрической аномалией, известное как *уравнение Кеплера*:

$$E(t)-\frac{180°}{\pi}e\sin E(t)=M(t), \qquad (4.7)$$

где

$$M(t)=360°\frac{(t-t_0)}{T}. \qquad (4.8)$$

Чтобы определить положение кометы на орбите в любой момент времени, нужно разрешить это уравнение относительно E, а затем вычислить x и y, подставив полученное значение в формулу (4.5). Эта процедура будет подробно описана в разделе, посвященном решению уравнения Кеплера.

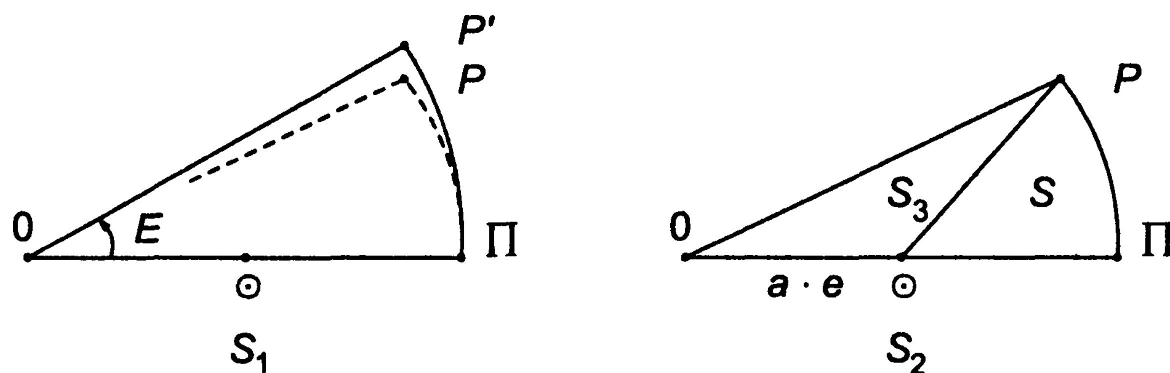


Рис. 4.4. Метод определения площади S эллиптического сектора

До сих пор период обращения по орбите рассматривался нами как независимый параметр. В действительности он связан с величиной большой полуоси орбиты третьим законом Кеплера.

$$\frac{a^3}{T^2}=\frac{GM_\odot}{4\pi^2}, \text{ где } GM_\odot=1{,}32712\cdot10^{20}\,\mathrm{m^3s^{-2}}. \qquad (4.9)$$

Константа в правой части уравнения содержит произведение гравитационной постоянной на массу Солнца $GM_\odot$. Однако стандартные единицы измерения, в которых выражены эти величины — метры и секунды, — редко используются в астрономических вычислениях. На практике чаще пользуются сутками $(1\,\mathrm{d}=86400\,\mathrm{s})$ и астрономическими единицами (1 а. е. ≈ 149,59787 млн км). Первоначально астрономическая единица определялась как длина большой полуоси земной орбиты. Однако позднее МАС принял решение, что она должна определяться из условия, чтобы произведение гравитационной постоянной на солнечную массу было равно фиксированной величине

$$GM_\odot=k^2(1\,\text{а.е.})^3\,\text{сут.}^{-2}, \text{ где } k=0{,}01720209895;\ k^2\approx2{,}959122083\cdot10^{-4}.$$

Здесь k — гауссова гравитационная постоянная. Хотя, на первый взгляд, такое определение кажется странным, у него есть одно важное преимущество: величина астрономической единицы не зависит от возможных вековых изменений диаметра земной орбиты.

После того как положение кометы на орбите определено, необходимо вычислить ее скорость. Для этого можно воспользоваться следующими формулами:

$$\begin{aligned} \dot{x} &= -\sqrt{\frac{GM_{\odot}}{a}}\frac{\sin E}{1-e\cos E}, \\ \dot{y} &= +\sqrt{\frac{GM_{\odot}\left(1-e^2\right)}{a}}\frac{\cos E}{1-e\cos E}. \end{aligned} \tag{4.10}$$

Для гиперболических орбит можно получить уравнения, аналогичные только что рассмотренным, но вместо тригонометрических функций в них будут использоваться гиперболические: $\sinh x = (e^x - e^{-x})/2$ и $\cosh x = (e^x + e^{-x})/2$. В результате имеем:

$$\begin{aligned} x &= r\cos\nu = |a|(e - \mathrm{ch}H), \\ y &= r\sin\nu = |a|\sqrt{e^2-1}\,\mathrm{sh}H \end{aligned} \tag{4.11}$$

и

$$\begin{aligned} \dot{x} &= -\sqrt{\frac{GM_{\odot}}{|a|}}\frac{\mathrm{sh}H}{e\,\mathrm{ch}H-1}, \\ \dot{y} &= +\sqrt{\frac{GM_{\odot}(e^2-1)}{|a|}}\frac{\mathrm{ch}H}{e\,\mathrm{ch}H-1}. \end{aligned} \tag{4.12}$$

Параметр H, который соответствует эксцентрической аномалии в случае эллиптических орбит, позволяет нам записать уравнение Кеплера в модифицированной форме

$$e\,\mathrm{sh}H - H = M_h = \sqrt{\frac{GM_{\odot}}{|a|^3}}\cdot(t-t_0). \tag{4.13}$$

Для параболических орбит координаты x и y выражаются через величину $\mathrm{tg}(\nu/2)$:

$$\begin{aligned} x &= r\cos\nu = q\cdot\left(1-\mathrm{tg}^2\left(\frac{\nu}{2}\right)\right), \\ y &= r\sin\nu = 2q\cdot\mathrm{tg}\left(\frac{\nu}{2}\right). \end{aligned} \tag{4.14}$$

Как уже отмечалось, q обозначает перигелийное расстояние. Истинную аномалию ν можно получить из уравнения Баркера

$$\operatorname{tg}\left(\frac{\nu}{2}\right)+\frac{1}{3}\operatorname{tg}^3\left(\frac{\nu}{2}\right)=\sqrt{\frac{GM_\odot}{2q^3}}\,(t-t_0). \qquad (4.15)$$

Это уравнение третьего порядка относительно tg $(\nu/2)$. В отличие от уравнения Кеплера его можно аналитически разрешить относительно ν. Вводя обозначения

$$A=\frac{3}{2}\sqrt{\frac{GM_\odot}{2q^3}}(t-t_0) \quad \text{и} \quad B=\sqrt[3]{A+\sqrt{A^2+1}},$$

получаем

$$\operatorname{tg}(\nu/2)=B-1/B, \quad \nu=2\operatorname{arctg}(B-1/B).$$

4.3. Численное решение уравнения Кеплера

Уравнение Кеплера не решается аналитически, то есть эксцентрическую аномалию невозможно явно представить, как функцию средней аномалии. Тем не менее его вполне можно решить численно.

Прежде всего выразим среднюю аномалию M как функцию времени t. Для орбит с малыми эксцентриситетами она мало отличается от эксцентрической аномалии E, и это позволяет использовать ее в качестве начального приближения E_0 в последующем итерационном процессе. Нетрудно заметить, что решение уравнения Кеплера сводится к отысканию корней функции

$$f(E)=E-\frac{180°}{\pi}e\sin E-M(t).$$

Для этого можно воспользоваться *методом Ньютона* (рис. 4.5). Проведем в точке x, близкой к корню, касательную к графику. Ее наклон составит $f'(x)$, а точка пересечения с осью x даст новое приближение для корня уравнения:

$$\hat{x}=x-\frac{f(x)}{f'(x)}.$$

В случае уравнения Кеплера итерационная формула имеет вид

$$\hat{E}=E-\frac{E-(180°/\pi)\cdot e\cdot\sin E-M(t)}{1-e\cdot\cos E}.$$

Здесь значение E надо задавать в градусах. При использовании радианной меры углов множитель $180°/\pi$ следует заменить единицей. Для орбит, близких к круговым, обычно достаточно трех итераций, чтобы получить решение с точностью до десяти знаков после запятой. Однако при сильно вытянутой орбите использовать метод Ньютона надо с большой осторожностью. Читатели могут сами посмотреть, что получится в случае орбиты с эксцентриситетом $e=0{,}99$, если в качестве начального приближения взять $E_0=M=5°$. В таких случаях следует использовать другое начальное значение, а именно $E_0=\pi=180°$. Такое начальное приближение исполь-

зуется в функции EccAnom для орбит эксцентриситетами больше, чем $e = 0{,}8$. Итерации повторяются до тех пор, пока найденное приближенное значение корня не будет удовлетворять уравнению Кеплера с точностью лишь на два десятичных разряда меньшей, чем точность типа double. Значение точности машинных вычислений можно получить, вызвав функцию numeric_limits <double>::epsilon из модуля <limits> стандартной библиотеки C++.

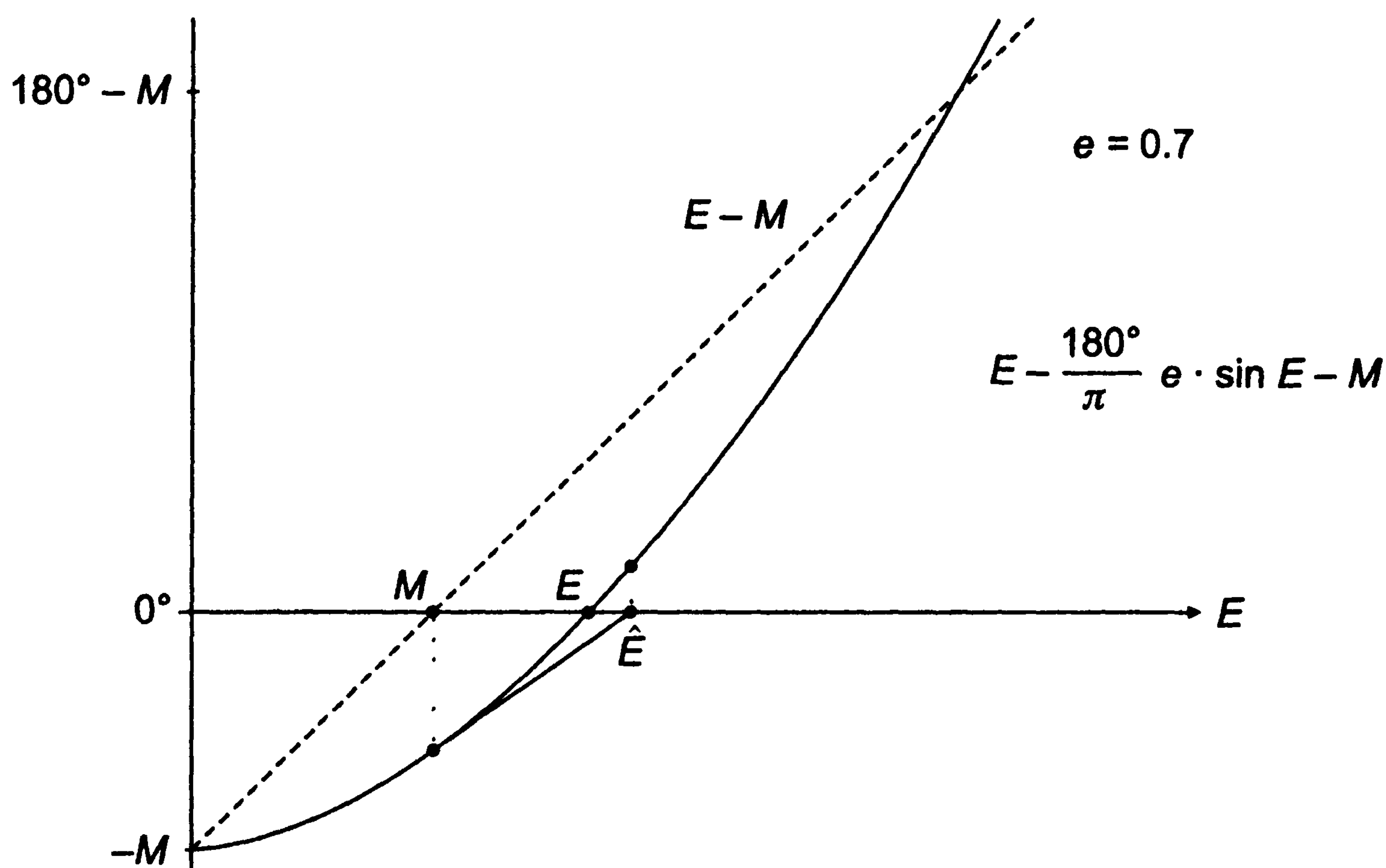


Рис. 4.5. Решение уравнения Кеплера методом Ньютона

```
const double eps_mach = numeric_limits<double>::epsilon();
//------------------------------------------------------------------------
// EccAnom: вычисляет эксцентрическую аномалию для эллиптической орбиты
//   M          Средняя аномалия [рад]
//   e          Эксцентриситет орбиты [0,1[
// <return>:  Эксцентрическая аномалия [рад]
//------------------------------------------------------------------------

double EccAnom (double M, double e)
{
  const int maxit = 15;
  const double eps = 100.0*eps_mach;
  int    i=0;
  double E, f;
  // Начальное приближение
  M = Modulo(M, 2.0*pi); if (e<0.8) E=M;
  else E=pi;
  // Итерация
  do {
    f = E - e*sin(E) - M;
    E = E - f / ( 1.0 - e*cos(E) );
    ++i;
    if (i==maxit) {
      cerr << " Convergence problems in EccAnom" << endl;
      break;
    }
  }
```

```
  while (fabs(f) > eps);
  return E;
}
```

Теперь для вычисления эксцентрической аномалии достаточно один раз обратиться к этой функции. Остальные шаги, необходимые для вычисления положения кометы на эллиптической орбите, реализованы в функции Ellip:

```
//---------------------------------------------------------------------------
// Ellip: вычисляет векторы положения и скорости для эллиптической орбиты
//   GM      Произведение постоянной тяготения и центральной массы [a.e.^3*d^-2]
//   M       Средняя аномалия [рад]
//   a       Большая полуось орбиты [a.e.]
//   e       Эксцентриситет орбиты (<1)
//   r       Вектор положения плоскости орбиты [a e.]
//   v       Вектор скорости относительно плоскости орбиты [a.e./d]
//---------------------------------------------------------------------------
void Ellip ( double GM, double M, double a, double e,
             Vec3D& r, Vec3D& v )
{
  double k, E, cosE,sinE, fac, rho;
  k  = sqrt(GM/a),
  E  = EccAnom(M,e);
  cosE = cos(E),
  sinE = sin(E);
  fac = sqrt ( (1.0-e)*(1.0+e) );
  rho = 1.0 - e*cosE;
  r = Vec3D (a*(cosE-e),  a*fac*sinE,     0.0);
  v = Vec3D (-k*sinE/rho, k*fac*cosE/rho, 0.0),
}
```

Аналогичным образом уравнение Кеплера решается и для гиперболических орбит. В этом случае шаг итерации принимает вид

$$\hat{H} = H - \frac{e \cdot \mathrm{sh}H - H - M_\mathrm{h}(t)}{e \cdot \mathrm{ch}H - 1}.$$

В качестве начального приближения рекомендуется брать

$$H_0 = \begin{cases} +\ln(+1{,}8 + 2M_\mathrm{h}/e) & \text{для } M_\mathrm{h} \ge 0, \\ -\ln(+1{,}8 - 2M_\mathrm{h}/e) & \text{для } M_\mathrm{h} < 0. \end{cases}$$

```
//---------------------------------------------------------------------------
// HypAnom. вычисляет эксцентрическую аномалию для гиперболической орбиты
//   Mh      Средняя аномалия [рад]
//   e       Эксцентриситет орбиты (>1)
// <return>: Эксцентрическая аномалия [рад]
//---------------------------------------------------------------------------
double HypAnom (double Mh, double e)
{
  const int maxit = 15;
  const double eps = 100.0*eps_mach;
  int    i=0,
  double H, f,
  // Начальное приближение
  H = log (2.0*fabs(Mh)/e + 1.8);
```

```
  if (Mh < 0.0)  H = -H.
  // Итерация
  do {
    f = e*sinh(H) - H - Mh.
    H = H - f / ( e*cosh(H) - 1.0 ):
    ++i:
    if (i==maxit) {
      cerr << " Convergence problems in HypAnom" << endl:
      break:
    }
  }
  while ( fabs(f) > eps*(1.0+fabs(H+Mh)) ):
  return H:
}
```

Прямоугольные координаты кометы, двигающейся по гиперболической орбите, вычисляет функция Hyperb по формулам, приведенным в предыдущем разделе. В отличие от функции Ellip вместо средней аномалии M_h она получает на входе момент прохождения перигелия и момент времени, для которого производится расчет. Это различие вызвано тем, что функция Ellip используется также при расчете движения астероидов и планет, для которых, в отличие от комет, момент прохождения перигелия редко используется в качествеэлемента орбиты. Вместо него обычно известны средняя аномалия M_{Ep} и среднее суточное движение n на определенную эпоху t_{Ep}. Среднюю аномалию $M = M_{Ep} + n(t - t_{Ep})$ на заданный момент времени t можно непосредственно использовать в качестве входных данных для функции Ellip. Гиперболические орбиты встречаются практически только у комет, для которых момент прохождения перигелия всегда приводится в качестве одного из элементов орбиты.

```
//------------------------------------------------------------------------------
// Hyperb: вычисляет положение и скорость для гиперболической орбиты
//   GM      Произведение постоянной тяготения и центральной массы [а.е ^3*d^-2]
//   t0      Момент прохождения перигелия
//   t       Расчетный момент
//   a       Большая полуось орбиты [а.е.]
//   e       Эксцентриситет орбиты (>1)
//   r       Вектор положения относительно плоскости орбиты [а.е.]
//   v       Вектор скорости относительно плоскости орбиты [а.е./d]
// Примечание: t0 и t отсчитываются от эпохи J2000 в юлианских столетиях
//------------------------------------------------------------------------------
void Hyperb ( double GM. double t0. double t. double a. double e. Vec3D& r. Vec3D& v )
{
  double k. Mh. H. coshH. sinhH. rho. fac:
  a  = fabs(a):
  k  = sqrt(GM/a).
  Mh = k*(t-t0)/a:
  H  = HypAnom(Mh.e):
  coshH = cosh(H).
  sinhH = sinh(H).
  fac = sqrt ( (e+1.0)*(e-1.0) ):
  rho = e*coshH - 1 0:
  r = Vec3D (a*(e-coshH).  a*fac*sinhH.     0.0):
  v = Vec3D (-k*sinhH/rho. k*fac*coshH/rho. 0.0):
}
```

4.4. Околопараболические орбиты

В предыдущем разделе мы уже обратили внимание на то, что расчет эллиптических и гиперболических орбит с эксцентриситетом, близким к единице, сталкивается с определенными трудностями при малых значениях аномалий. Этих трудностей можно избежать, воспользовавшись тем, что такие орбиты вблизи перигелия очень похожи на параболические. Мы применим метод, впервые предложенный Карлом Штумпфом.

Из уравнения Кеплера, записанного в другой форме

$$E(t) - e\sin E(t) = \sqrt{\frac{GM_\odot}{a^3}} \cdot (t - t_0),$$

и соотношений

$$a = q / (1-e) \quad \text{и} \quad c_3 = (E - \sin E) / E^3$$

получаем:

$$(1-e) \cdot E + e \cdot c_3(E) \cdot E^3 = \sqrt{GM_\odot \left(\frac{1-e}{q}\right)^3} \cdot (t - t_0).$$

Введя новую переменную

$$U = \sqrt{\frac{3e \cdot c_3(E)}{1-e}} \cdot E,$$

запишем это уравнение в форме

$$U + \frac{1}{3}U^3 = \sqrt{6ec_3(E)} \cdot \sqrt{\frac{GM_\odot}{2q^3}} \cdot (t - t_0).$$

Полученное уравнение имеет ту же форму, что и уравнение Баркера, и может быть разрешено относительно U при известной правой части. Вводя обозначения $c_1(E) = \sin E / E$ и $c_2(E) = (1 - \cos E)/E^2$, получаем:

$$\begin{aligned} x &= r\cos\nu = \frac{q}{1-e}(\cos E - e) &&= q \cdot \left(1 - \left[\frac{2c_2}{6ec_3}\right] U^2\right), \\ y &= r\sin\nu = \frac{q}{1-e}\sqrt{1-e^2}\sin E &&= 2q \cdot \sqrt{\frac{1+e}{2e} \cdot \left[\frac{1}{6c_3}\right]} \cdot c_1 \cdot U, \\ r & &&= q \cdot \left(1 + \left[\frac{2c_2}{6c_3}\right] U^2\right). \end{aligned} \tag{4.16}$$

Далее нам понадобятся выражения для компонентов скорости кометы. Их можнополучить, дифференцируя уравнения (4.16):

$$\dot{x} = -\sqrt{\frac{GM_\odot}{q(1+e)}} \left(\frac{y}{r}\right), \tag{4.17}$$

$$\dot{y} = +\sqrt{\frac{GM_\odot}{q(1+e)}}\left(\frac{x}{r}+e\right).$$

Наконец, приведем ради полноты соотношение между U и истинной аномалией ν

$$\operatorname{tg}\left(\frac{\nu}{2}\right) = \left[\sqrt{\frac{1+e}{3ec_3}}\cdot\frac{c_2}{c_1}\right]\cdot U.$$

В приведенных уравнениях члены, заключенные в квадратные скобки, стремятся к 1 при $E \to 0$ и $e \to 1$. В этом предельном случае мы приходим к уравнениям для параболической орбиты, которые мы уже обсуждали.

Конечно, новая форма уравнений не позволит нам обойтись при решении без использования последовательных приближений. Важно, однако, что теперь итерационный процесс не сталкивается с трудностями. В качестве начального приближения мы выберем $E \approx 0$ ($c_3(E) \approx 1/6$) и определим соответствующее значение U, решая уравнение Кеплера–Баркера

$$U = B - 1/B,$$

где

$$B = \sqrt[3]{A + \sqrt{A^2+1}} \quad \text{и} \quad A = \frac{3}{2}\sqrt{6ec_3(E)}\sqrt{\frac{GM_\odot}{2q^3}}\cdot(t-t_0).$$

Отсюда получаем уточненные значения:

$$\hat{E} = U\cdot\sqrt{\frac{1-e}{3e\cdot c_3(E)}} \quad \text{и} \quad c_3(\hat{E}) = \frac{\hat{E}-\sin\hat{E}}{\hat{E}^2},$$

которые можно использовать для определения уточненного значения $\hat{U}$. Эти итерации следует повторять до тех пор, пока изменения U не станут меньше требуемой точности. В ходе вычислений очень важно корректно вычислять значения функций $c_1(E), c_2(E)$ и $c_3(E)$ при малых значениях E. Поскольку они представляют собой отношения почти равных величин, их нельзя определять путем прямого вычисления соответствующих тригонометрических функций. Вместо этого следует пользоваться соответствующими разложениями в ряды Тейлора:

$$\begin{aligned} c_1(E) &= \frac{\sin E}{E} = 1 - \frac{E^2}{3!} + \frac{E^4}{5!} - \ldots = \sum_{n=1}^{\infty}\alpha_n, \\ c_2(E) &= \frac{1-\cos E}{E^2} = \frac{1}{2!} - \frac{E^2}{4!} + \frac{E^4}{6!} - \ldots = \sum_{n=1}^{\infty}\beta_n, \\ c_3(E) &= \frac{E-\sin E}{E^3} = \frac{1}{3!} - \frac{E^2}{5!} + \frac{E^4}{7!} - \ldots = \sum_{n=1}^{\infty}\gamma_n. \end{aligned} \tag{4.18}$$

Все три ряда содержат только четные степени E и поэтому быстро сходятся. Заметим, что члены этих рядов удобно вычислять, пользуясь следующей рекурсивной процедурой:

$$\alpha_1 = 1,$$
$$\beta_n = \alpha_n \cdot \frac{1}{2n}, \quad \gamma_n = \beta_n \cdot \frac{1}{2n+1}, \quad \alpha_{n+1} = -E^2 \cdot \gamma_n \quad (n=1, \ldots).$$

Именно так работает функция Stumpff, которая суммирует члены рядов (4.18) до тех пор, пока они не станут меньше заданной точности eps.

```
//---------------------------------------------------------------------------
// Stumpff: вычисляет значения функций Штумпфа C1, C2 и C3
//   E2       Квадрат эксцентрической аномалии (E2=E*E) [рад^2]
//   c1       Значение функции C1 = sin(E)/E
//   c2       Значение функции C2 = (1-cos(E))/(E*E)
//   c3       Значение функции C3 = (E-sin(E))/(E^3)
//---------------------------------------------------------------------------
void Stumpff (double E2, double& c1, double& c2, double& c3)
{
  const double eps = 100.0*eps_mach;
  double      n, add;
  c1 = c2 = c3 = 0.0;
  add = n = 1.0;
  do {
    c1 += add;  add /= (2.0*n);
    c2 += add;  add /= (2.0*n+1.0);
    c3 += add;  add *= -E2;
    n += 1.0;
  }
  while (fabs(add) >= eps);
}
```

Остальные вычисления для параболических и близких к параболическим орбит выполняет функция Parab.

```
//---------------------------------------------------------------------------
// Parab: вычисляет векторы положения и скорости для параболических
//        и близких к параболическим орбит
//   GM       Произведение постоянной тяготения и центральной массы [a.e.^3*d^-2]
//   t0       Момент прохождения перигелия
//   t        Момент, для которого производятся вычисления
//   q        Перигелийное расстояние [a.e.]
//   e        Эксцентриситет орбиты (>1)
//   r        Вектор положения относительно плоскости орбиты [a.e.]
//   v        Вектор скорости относительно плоскости орбиты [a.e./d]
// Примечание: t0 и t отсчитываются от эпохи J2000 в юлианских столетиях
//---------------------------------------------------------------------------
void Parab ( double GM, double t0, double t, double q, double e, Vec3D& r, Vec3D& v )
{
  const int maxit = 15;
  const double eps = 100.0*eps_mach;
  int     i=0;
  double  E2=0.0;
  double  E20, fac, c1, c2, c3, k, tau, A, B, u, u2, R;
  fac = 0.5*e;
  k   = sqrt(GM / (q*(1.0+e)));
  tau = sqrt(GM)*(t-t0);
  do {
    ++i;
    E20 = E2;
    A = 1.5*sqrt(fac/(q*q*q))*tau;
```

```
    B = pow ( sqrt(A*A+1.0)+A, 1 0/3 0 ),
    u  = B - 1.0/B;
    u2 = u*u;
    E2 = u2*(1.0-e)/fac;
    Stumpff (E2, c1,c2,c3);
    fac = 3.0*e*c3;
    if (i == maxit) {
      cerr << " Convergence problems in Parab" << endl;
      break;
    }
  }
  while (fabs(E2-E20) >= eps);
  R  = q * ( 1.0 + u2*c2*e/fac );
  r = Vec3D (q*(1.0-u2*c2/fac), q*sqrt((1 0+e)/fac)*u*c1, 0.0);
  v = Vec3D (-k*r[y]/R, k*(r[x]/R+e), 0 0);
}
```

4.5. Векторы Гаусса

Рассмотренные нами функции позволяют вычислить положение кометы в плоскости ее орбиты. Для получения эклиптических координат кометы нужно прежде всего определить ориентацию орбиты относительно эклиптики. Для этого служат еще три элемента орбиты (рис. 4.6).

- i Наклонение — это угол, под которым плоскость орбиты пересекается с плоскостью эклиптики. Если наклонение превышает 90°, это означает, что комета имеет обратное направление орбитального движения, то есть движется по орбите в направлении, противоположном движению планет.
- ☊ *Долгота восходящего узла* определяет угол между точкой весеннего равноденствия и точкой орбиты, в которой комета пересекает эклиптику, переходя из южного полушария в северное.
- ω Аргумент перигелия — угол между направлением на восходящий узел и направлением на ближайшую к Солнцу точку орбиты.

Часто вместо последнего элемента ω используют *долготу перигелия* ($\varpi = ☊ + \omega$), определяемую как сумма долготы восходящего узла и аргумента перигелия.

Поскольку положение точки весеннего равноденствия медленно меняется вследствие прецессии, элементы орбиты должны сопровождаться указанием эпохи, к которой они относятся. Например, равноденствие 1950,0 означает, что элементы орбиты даны относительно положения эклиптики и точки весеннего равноденствия в 1950 году.

Для получения формул преобразования координат начнем с введения перифокальной системы координат. За основу в ней приняты плоскость орбиты и линия апсид. В этой системе координат точка орбиты с истинной аномалией ν и расстоянием от Солнца r имеет координаты

$$\tilde{\boldsymbol{r}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{pmatrix} r\cos\nu \\ r\sin\nu \\ 0 \end{pmatrix}.$$

Здесь тильда служит для того, чтобы отличать перифокальные координаты от эклиптических (x, y, z).

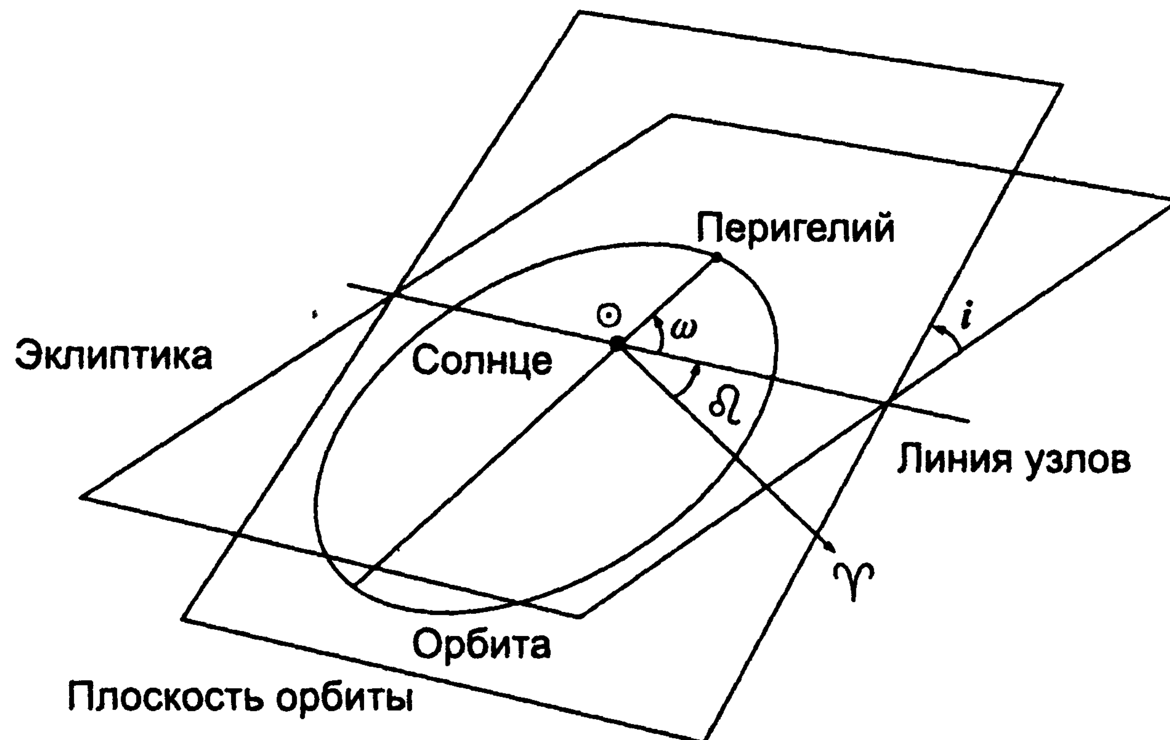


Рис. 4.6. Элементы орбиты i, ☊ и ω

В результате решения уравнений Кеплера и Баркера мы получаем координаты кометы в системе (x'', y'', z''), основная плоскость $x'' - y''$ которой совпадает с плоскостью орбиты, а ось x'' направлена на восходящий узел. Перифокальная система получается поворотом этой системы координат вокруг оси $\tilde{z}/z''$ на угол ω в положительном направлении (рис. 4.7). Для координат точки орбиты получаем

$$\boldsymbol{r}'' = \boldsymbol{R}_z(-\omega)\tilde{\boldsymbol{r}} = \begin{pmatrix} +\cos\omega & -\sin\omega & 0 \\ +\sin\omega & +\cos\omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r\cos\nu \\ r\sin\nu \\ 0 \end{pmatrix} = \begin{pmatrix} r\cos u \\ r\sin u \\ 0 \end{pmatrix},$$

где *аргумент долготы u* представляет собой сумму аргумента перигелия и истинной аномалии

$$u = \omega + \nu. \qquad (4.19)$$

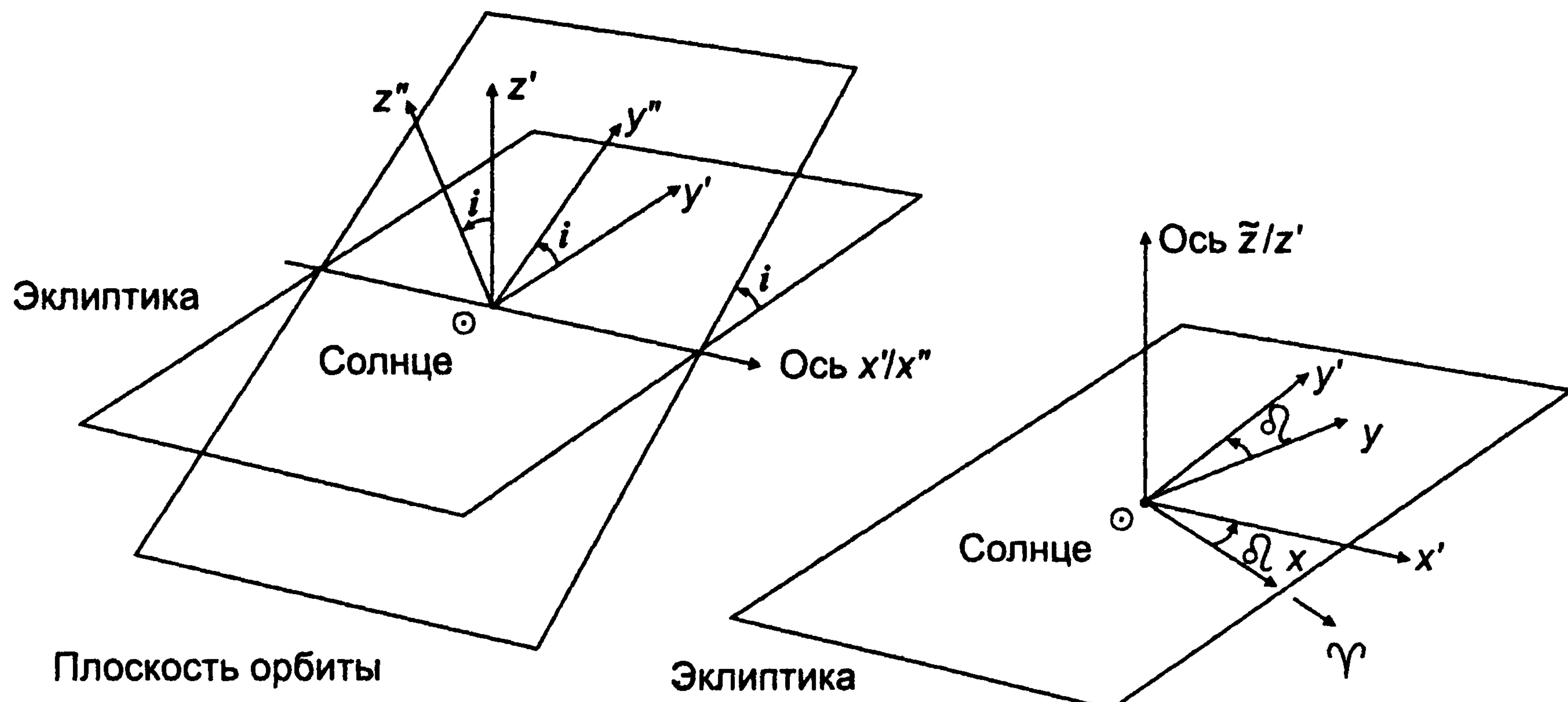


Рис. 4.7. Преобразование системы координат

Теперь рассмотрим систему координат (x', y', z'), где ось x' также указывает на восходящий узел, но плоскость x'–y' совпадает с плоскостью эклиптики. Эти две системы координат наклонены друг по отношению к другу на угол i:

$$r' = R_x(-i)r'' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & +\cos i & -\cos i \\ 0 & +\sin i & +\cos i \end{pmatrix} \begin{pmatrix} r\cos u \\ r\sin u \\ 0 \end{pmatrix} = \begin{pmatrix} r\cos u \\ r\sin u\cos i \\ r\sin u\sin i \end{pmatrix}.$$

После этого преобразования координаты точки уже будут отнесены к плоскости эклиптики, но пока еще связаны с восходящим узлом орбиты. Нам остается выполнить последнее преобразование и перейти к системе координат (x, y, z), которая обычным образом соотносится с эклиптикой и точкой весеннего равноденствия

$$r = R_z(-\Omega)r' = \begin{pmatrix} +\cos\Omega & -\sin\Omega & 0 \\ +\sin\Omega & +\cos i & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} r\cos u \\ r\sin u\cos i \\ r\sin u\sin i \end{pmatrix}.$$

Объединяя все эти уравнения, получаем формулу

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = r\begin{pmatrix} \cos u\cos\Omega - \sin u\cos i\sin\Omega \\ \cos u\sin\Omega - \sin u\cos i\cos\Omega \\ \sin u\ \sin i \end{pmatrix}, \tag{4.20}$$

которая дает возможность определить эклиптические координаты (x, y, z) по заданным расстоянию r и истинной аномалии ν.

Описанные повороты можно представить одной матрицей

$$U = R_z(-\Omega)R_x(-i)R_z(-\omega), \tag{4.21}$$

позволяющей в особенно компактной форме записать преобразование из перифокальной системы координат в эклиптическую

$$r = U\tilde{r} \quad \dot{r} = U\dot{\tilde{r}}. \tag{4.22}$$

Пользоваться этими соотношениями удобно в тех случаях, когда требуется преобразовывать не только положения, но и скорости, а также при работе с рядами положений кометы в разные моменты времени. Элементы матрицы вычисляются однократно в начале расчета, поскольку они зависят только от элементов орбиты, но не от момента времени.

Столбцы (P, Q, R) матрицы U задают три взаимно перпендикулярных вектора, называемых векторами Гаусса. Первые два из них

$$P = \begin{pmatrix} +\cos\omega\cos\Omega - \sin\omega\cos i\ \sin\Omega \\ +\cos\omega\ \sin\Omega - \sin\omega\cos i\cos\Omega \\ +\sin\omega\ \sin i \end{pmatrix} \tag{4.23}$$

и

$$\boldsymbol{Q} = \begin{pmatrix} -\sin\omega\cos\Omega - \cos\omega\cos i\,\sin\Omega \\ -\sin\omega\,\sin\Omega - \cos\omega\cos i\cos\Omega \\ +\cos\omega\,\sin i \end{pmatrix} \qquad (4.24)$$

лежат в плоскости орбиты и направлены соответственно к перигелию и к точке с истинной аномалией $\nu = 90°$, а третий вектор

$$\boldsymbol{R} = \begin{pmatrix} +\sin i\,\sin\Omega \\ -\sin i\cos\Omega \\ +\cos i \end{pmatrix} \qquad (4.25)$$

перпендикулярен плоскости орбиты и поэтому зависит только от i и Ω, но не от аргумента перигелия ω.

Функция GaussVec вычисляет векторы Гаусса по заданным элементам орбиты и представляет их в форме матрицы размером 3×3, возвращая объект класса Mat3D. В соответствии с формулой (4.21) полная матрица преобразования строится как произведение матриц соответствующих поворотов. По сравнению с покомпонентным вычислением это позволяет значительно компактнее записать соответствующие вычисления.

```
//------------------------------------------------------------------------------
// GaussVec: вычисляет матрицу преобразования координат от системы, связанной
//           с плоскостью орбиты, к эклиптической системе координат
//   Omega    Долгота восходящего узла орбиты [рад]
//   i        Наклонение орбиты к эклиптике [рад]
//   omega    Аргумент перигелия [рад]
// <return>:  Матрица преобразования, состоящая их трех векторов Гаусса P, Q и R
//------------------------------------------------------------------------------
Mat3D GaussVec (double Omega, double i, double omega)
{
  return R_z(-Omega) * R_x(-i) * R_z(-omega);
}
```

Пользуясь этой функцией, нетрудно доработать рассмотренные в предыдущем разделе процедуры расчета положения кометы для различных типов орбит так, чтобы сразу получать гелиоцентрические эклиптические координаты.

```
//------------------------------------------------------------------------------
// Kepler: вычисляет векторы положения и скорости относительно эклиптики
//         для кеплеровских орбит
//   GM       Произведение постоянной тяготения и центральной массы [a.e.^3*d^-2]
//   t0       Момент прохождения перигелия
//   t        Расчетный момент
//   q        Перигелийное расстояние [a.e.]
//   e        Эксцентриситет орбиты
//   PQR      Преобразование плоскость орбиты -> эклиптика (векторы Гаусса)
//   r        Вектор положения относительно плоскости орбиты [a.e.]
//   v        Вектор скорости относительно плоскости орбиты [a.e./d]
// Примечание: t0 и t отсчитываются от эпохи J2000 в юлианских столетиях
//------------------------------------------------------------------------------
void Kepler ( double GM, double t0, double t,
              double q, double e, const Mat3D& PQR,
              Vec3D& r, Vec3D& v )
```

```
{
  const double M0  = 0.1; // [rad]
  const double eps = 0 1;
  double M, delta, tau, invax;
  Vec3D  r_orb,v_orb;
  delta = fabs(1.0-e);
  invax = delta / q;
  tau   = sqrt(GM)*(t-t0);
  M     = tau * sqrt(invax*invax*invax);
  if ( (M < M0) && (delta < eps) )
    Parab (GM, t0,t, q,e, r_orb, v_orb);
  else if ( e < 1.0 )
    Ellip (GM, M, 1.0/invax, e, r_orb, v_orb),
  else
    Hyperb (GM, t0,t, 1.0/invax, e, r_orb, v_orb),
  r = PQR*r_orb;
  v = PQR*v_orb;
}
```

4.6. Световая задержка

Свету, идущему от кометы, необходимо время для того, чтобы достичь Земли. В зависимости от расстояния до кометы на это может потребоваться от нескольких минут до нескольких часов. Пока свет идет к Земле, комета продолжает свое движение, и поэтому ее видимое положение оказывается немного позади реального геометрического. Скорость света намного больше скоростей, характерных для комет, и, следовательно, учет световой задержки приводит лишь к незначительным поправкам— от нескольких секунд до примерно одной минуты дуги. Это позволяет нам использовать приближенные расчеты, делая учет световой задержки относительно простой процедурой. Для вычисления положения $\boldsymbol{r}(t)$, в котором комета будет видна в момент времени t, определим сначала ее гелиоцентрические координаты $\boldsymbol{r}_k(t)$ и геоцентрические координаты Солнца $\boldsymbol{r}_\odot(t)$. Геометрическое расстояние кометы от Земли можно тогда вычислить по формуле

$$\Delta_0 = |\boldsymbol{r}_0| = |\boldsymbol{r}_\odot(t) + \boldsymbol{r}_k(t)| = \sqrt{(x_\odot + x_k)^2 + (y_\odot + y_k)^2 + (z_\odot + z_k)^2}\,.$$

Эта величина лишь незначительно отличается от расстояния, реально проходимого светом:

$$\Delta = |\boldsymbol{r}_\odot(t) + \boldsymbol{r}_k(t')|, \qquad \text{где} \qquad t' = t - \frac{\Delta}{c},$$

которое можно вычислить только методом последовательных приближений, так как момент времени t', когда свет был испущен кометой, нельзя определить заранее. Таким образом, световая задержка составляет $\tau \approx \Delta_0 / c$, и гелиоцентрические координаты кометы в момент t' можно с хорошей точностью определить по формуле

$$\boldsymbol{r}_k(t') \approx \boldsymbol{r}_k(t) - \boldsymbol{v}_k(t) \cdot \frac{\Delta_0}{c}.$$

С использованием этих *запаздывающих* гелиоцентрических координат геоцентрический вектор наблюдаемого положения кометы определяется как

$$\boldsymbol{r} \approx \boldsymbol{r}_{\odot}(t) + \left\{ \boldsymbol{r}_k(t) - \boldsymbol{v}_k(t) \cdot \frac{\Delta_0}{c} \right\}. \quad (4.26)$$

Множитель $1/c$ в используемых нами единицах измерения (а. е. и сутки) имеет численное значение

$$1/c = 0^{\mathrm{d}}\!.00578/\text{а. е.}$$

Координаты кометы, с учтенной поправкой на световую задержку, называются *астрометрическими* координатами. Они годятся для непосредственного сопоставления положения кометы с координатами звезд, взятыми из звездного атласа или каталога. При этом надо заметить, что в качестве геоцентрического расстояния до кометы обычно приводят геометрическое расстояние Δ_0, а не путь Δ, реально проходимый светом).

4.7. Программа Comet

Программа Comet рассчитывает эфемериды комет и астероидов на основе решения задачи двух тел. Возмущения, которые претерпевает орбита небесного тела под действием планет, не принимаются в расчет. Такое упрощение обычно вполне приемлемо для большинства практических целей. Однако, сравнивая наши расчеты с результатами, приводимыми в других источниках, нужно иметь в виду, что последние могли и не использовать таких упрощений. Пользователю можно не беспокоиться, какого типа орбита кометы (эллиптическая, параболическая или гиперболическая), — программа Comet, учитывая значение эксцентриситета, сама выберет нужный метод вычислений. В частности, для сильно вытянутых эллиптических орбит с большими эксцентриситетами используется рассмотренный выше метод Штумпфа.

Программа Comet вычисляет гелиоцентрические эклиптические и геоцентрические экваториальные координаты. В обоих случаях выполняется учет прецессии для заданной эпохи, а в геоцентрические координаты также вносится световая поправка. Исправленные таким образом геоцентрические координаты называются астрометрическими. Их можно непосредственно сравнивать с положениями звезд, приводимыми в звездных каталогах, и использовать для нанесения объекта на звездную карту, составленную для той же эпохи.

Сравнивая полученные результаты с данными, приводимыми в другом источнике, обязательно проверьте, действительно ли в нем даются астрометрические координаты, отнесенные к той же эпохе, а также не являются ли приводимые координаты видимыми, то есть исправленными за нутацию и аберрацию.

```
//------------------------------------------------------------------------------
// Файл:        Comet.cpp
// Назначение: Вычисление невозмущенных эфемерид комет и астероидов
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//------------------------------------------------------------------------------
```

```
#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>
#include "APC_Const.h"
#include "APC_IO.h"
#include "APC_Kepler.h"
#include "APC_Math.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"
using namespace std;

//------------------------------------------------------------------------------
// GetElm: Читает элементы орбиты из входного файла
//   Filename  Имя входного файла с элементами орбиты
//   Mjd0      Момент прохождения перигелия в форме модифицированной юлианской даты
//   q         Перигелийное расстояние [а.е.]
//   e         Эксцентриситет
//   PQR       Матрица преобразования координат от плоскости орбиты к эклиптике
//   T_eqx0    Эпоха, для которой заданы элементы орбиты
//------------------------------------------------------------------------------
void GetElm ( char* Filename,
              double& Mjd0, double& q, double& e, Mat3D& PQR, double& T_eqx0 )
{
  int      year, month;
  double   Omega, i, omega;
  double   Year,day;
  ifstream inp;
  inp.open(Filename);   // Открываем файл с элементами орбиты
  // Читаем элементы орбиты и выполняем контрольный вывод
  cout << endl
       << " Orbital elements from file " << Filename << endl << endl;
  inp >> year >> month >> day; inp.ignore(81,'\n');
  Mjd0  = Mjd(year,month,int(day))+(day-int(day));
  cout << "  perihelion time (y m d) "
       << "  " << setprecision(2) << DateTime(Mjd0,DDd) << endl;
  inp >> q; inp.ignore(81,'\n');
  cout << "  perihelion distance (q) "
       << setprecision(7) << setw(14) << q << " AU" << endl;
  inp >> e; inp.ignore(81,'\n');
  cout << "  eccentricity (e)        "
       << setprecision(7) << setw(14) << e << endl;
  inp >> i; inp.ignore(81,'\n');
  cout << "  inclination (i)         "
       << setprecision(5) << setw(12) << i << " deg" << endl;
  inp >> Omega; inp.ignore(81,'\n');
  cout << "  long. of ascending node "
       << setprecision(5) << setw(12) << Omega << " deg" << endl;
  inp >> omega; inp.ignore(81,'\n');
  cout << "  argument of perihelion  "
       << setprecision(5) << setw(12) << omega << " deg" << endl;
  inp >> Year; inp.ignore(81,'\n');
  cout << "  equinox                 "
```

```
      << setprecision(2) << setw(9)  << Year << endl.
  inp.close():
  T_eqx0 = (Year-2000.0)/100.0:
  PQR = GaussVec(Rad*Omega. Rad*i. Rad*omega).
}
//--------------------------------------------------------------------------------
// GetEph· Запрашивает у пользователя начальную и конечную даты эфемериды.
//         величину шага и требуемую эпоху
//   MjdStart  Начальный момент эфемериды в форме модифицированной юлианской даты
//   Step      Шаг эфемериды в сутках
//   MjdEnd    Конечный момент эфемериды в форме модифицированной юлианской даты
//   T_eqx     Требуемая эпоха в юлианских столетиях. отсчитываемых от J2000
//--------------------------------------------------------------------------------
void GetEph (double& MjdStart. double& Step. double& MjdEnd. double& T_eqx)
{
  int    year. month. day;
  double hour. Year;
  cout << endl
       << " Begin and end of the ephemeris. " << endl << endl;
  // Запрашивает у пользователя начальную и конечную даты. шаг и эпоху
  cout << "  first date (yyyy mm dd hh.hhh)            ... ";
  cin >> year >> month >> day >> hour; cin.ignore(81.'\n');
  MjdStart = Mjd(year. month. day) + hour/24 0 .
  cout << "  final date (yyyy mm dd hh.hhh)            . . ";
  cin >> year >> month >> day >> hour; cin ignore(81.'\n');
  MjdEnd = Mjd(year. month. day) + hour/24.0 ;
  cout << "  step size (dd hh.hh)                      ... ".
  cin >> day >> hour; cin ignore(81.'\n');
  Step  = day + hour/24.0:
  cout << endl << " Desired equinox of the ephemeris (yyyy y)       ":
  cin >> Year; cin.ignore(81.'\n');
  T_eqx = (Year-2000.0)/100.0:
}

//--------------------------------------------------------------------------------
//
// Основная программа
//
//--------------------------------------------------------------------------------
void main(int argc. char* argv[]) {
  // Переменные
  int        n_line = 0.
  double     MjdStart. Step. MjdEnd. T_eqx.
  double     Mjd0. q. e. T_eqx0:
  Mat3D      PQR.
  double     Date. T.
  double     dist. fac.
  Vec3D      R_Sun. r_helioc. v_helioc. r_geoc. r_equ:
  char       InputFile[APC_MaxFilename]  = "":
  char       OutputFile[APC_MaxFilename] = "":
  bool       FoundInputfile  = false.
  bool       FoundOutputfile = false:
  ofstream   OutFile:

  // Заголовок
  cout << endl
       << " COMET: ephemeris calculation for comets and minor planets" << endl
```

```
     << "        (c) 1999 Oliver Montenbruck, Thomas Pfleger       " << endl
     << endl;

  // Поиск входного файла
  GetFilenames( argc, argv, "Comet.dat", InputFile, FoundInputfile,
                OutputFile, FoundOutputfile );
  // Прекратить выполнение программы, если входной файл не найден
  if ( !FoundInputfile ) {
    cerr << " Terminating program." << endl;
    exit(-1);
  }
  // Ввод элементов орбиты и данных пользователя
  GetElm ( InputFile, MjdO, q, e, PQR, T_eqx0 );
  GetEph ( MjdStart, Step, MjdEnd, T_eqx );
  PQR = PrecMatrix_Ecl(T_eqx0, T_eqx) * PQR;

  // Перенаправление вывода, если создается выходной файл
  if (FoundOutputfile) {
    OutFile.open(OutputFile);
    if (OutFile.is_open())
      cout = OutFile;
  }
  // Заголовок таблицы
  cout << endl << endl
     << "    Date      ET   Sun      l      b      r"
     << "       RA          Dec       Distance " << endl
     << setw(43) << " " << "   h  m  s      o  '  \"     (AU) " << endl;

  // Вычисление эфемериды
  Date = MjdStart;

  // Цикл по времени
  while ( Date < MjdEnd + Step/2 ) {
    // Геоцентрические эклиптические координаты Солнца, эпоха T_eqx
    T = ( Date - 51544 5 ) / 36525.0;
    R_Sun = PrecMatrix_Ecl(T, T_eqx) * SunPos(T);
    // Гелиоцентрические эклиптические координаты кометы, эпоха T_eqx
    Kepler (GM_Sun, MjdO, Date, q, e, PQR, r_helioc, v_helioc);
    // Геометрические геоцентрические координаты кометы
    r_geoc = r_helioc + R_Sun,
    // Световая поправка, первое приближение
    dist = Norm(r_geoc);
    fac  = 0 00578*dist;
    r_geoc = r_geoc - fac*v_helioc;
    // Экваториальные координаты
    r_equ = Ecl2EquMatrix(T_eqx) * r_geoc;
    // Вывод
    cout << DateTime(Date,HHh)
         << fixed << setprecision(1)
         << setw(7) << Deg*R_Sun[phi]
         << setw(7) << Deg*r_helioc[phi]
         << setw(6) << Deg*r_helioc[theta]
         << setprecision(3) << setw(7)  << r_helioc[r]
         << setprecision(1) << setw(12) << Angle(Deg*r_equ[phi]/15.0,DMMSSs)
         << "  " << showpos << setw(9)  << Angle(Deg*r_equ[theta],DMMSS)
         << noshowpos << setprecision(6) << setw(11) << dist
         << endl;
```

```
    ++n_line;
    if ( (n_line % 5) == 0 ) cout << endl; // insert line feed every 5 lines
    Date += Step; // Next time step
  }; // Конец цикла по времени

  if (OutFile.is_open()) OutFile.close();
}
```

В качестве примера использования программы Comet вычислим эфемериду кометы Галлея на период ее видимости при последнем возвращении (1P/1982 U1). Прежде всего нужно создать файл с элементами орбиты. Мы назовем его Comet.dat. В каждой строке текст, начинающийся с восклицательного знака, является комментарием и при желании может быть опущен. Комета Галлея в период ее последнего возвращения в 1985/1986 годах имела следующие элементы орбиты:

```
1986 2 9.43867  ! Момент прохождения перигелия (год месяц день доля дня)
   0 5870992    ! Перигелийное расстояние q [а.е.]
   0.9672725    ! Эксцентриситет e
 162.23932      ! Наклонение i [град.]
  58.14397      ! Долгота восходящего узла [град.]
 111.84658      ! Аргумент перигелия [град.]
1950 0          ! Эпоха для элементов орбиты
```

При запуске программа Comet выполняет контрольный вывод данных входного файла, а затем запрашивает у пользователя интервал времени, для которого выполняется расчет, шаг расчета и эпоху:

```
COMET· ephemeris calculation for comets and minor planets
       (c) 1999 Oliver Montenbruck, Thomas Pfleger

Using default input file Comet.dat Orbital elements from file Comet.dat

 perihelion time (y m d)   1986/02/09.44
 perihelion distance (q)      0.5870992 AU
 eccentricity (e)             0.9672725
 inclination (i)            162.23932 deg
 long. of ascending node     58.14397 deg
 argument of perihelion     111.84658 deg
 equinox                   1950.00
```

Мы хотим получить эфемериду на период с 15 ноября 1985 года по 1 апреля 1986 года с десятидневным шагом. В качестве эпохи выберем 2000,0. Граничные даты эфемериды задаются в форме: год, месяц, день и час с десятичными долями. Данные, вводимые пользователем, выделены курсивом:

```
Begin and end of the ephemeris:
 first date (yyyy mm dd hh.hhh)          ... 1985 11 15 00.0
 final date (yyyy mm dd hh.hhh)          ... 1986 04 01 00.0
 step size (dd hh.hh)                    ... 1000.0

 Desired equinox of the ephemeris (yyyy.y) ... 2000.0
```

Теперь программа Comet вычисляет эфемериду для указанных дат:

```
  Date      ET   Sun     l      b    r        RA          Dec   Distance
                                              h  m  s     o  '  "   (AU)
1985/11/15 00.0 232.8   56.9   0.6 1.720   4 00 40.3 +22 04 27 0.736822
```

```
1985/11/25 00.0 242.9   53.2   1.8 1.572   2 15 19.1 +18 24 11 0.623053
1985/12/05 00.0 253.0   48.7   3.2 1.422   0 28 47.5 +10 39 00 0.665665
1985/12/15 00.0 263.2   43.1   5.0 1.269  23 18 18.6 + 3 53 22 0.821652
1985/12/25 00.0 273.3   35.9   7.1 1.114  22 36 56 1 - 0 20 05 1.019722

1986/01/04 00 0 283.5   26.2   9.8 0.961  22 10 36.0 - 3 00 26 1.216950
1986/01/14 00.0 293.7   12.8  13.0 0.814  21 50 58.9 - 4 58 44 1.388260
1986/01/24 00.0 303.9  353 4  16.2 0.687  21 33 26.2 - 6 47 36 1 511970
1986/02/03 00.0 314.1  326.3  17 7 0.604  21 15 33.4 - 8 49 26 1.563479
1986/02/13 00.0 324 2  294 8  14.9 0.592  20 57 11.7 -11 15 58 1.520207

1986/02/23 00.0 334 3 267.2    8.7 0.658  20 39 33.7 -14 09 18 1.383204
1986/03/05 00.0 344 3 247.1    2.6 0.775  20 22 12.2 -17 39 45 1.179039
1986/03/15 00.0 354.3 232.8   -1.9 0.918  20 00 46.1 -22 28 16 0.936979
1986/03/25 00.0   4.2 222.5   -5.2 1.070  19 22 34.1 -30 13 41 0.685469
1986/04/04 00.0  14.1 214.7   -7.5 1 225  17 41 16.2 -42 56 51 0.475274
```

Первые две колонки содержат дату и время. В колонке, озаглавленной Sun, приводится геоцентрическая эклиптическая долгота Солнца. Эту информацию можно использовать для того, чтобы оценить элонгацию кометы (или астероида) и понять, когда объект доступен для наблюдения. Следующие три колонки (озаглавленные l, b и r) содержат гелиоцентрические эклиптические координаты кометы. Далее приводятся геоцентрические прямое восхождение и склонение. Наконец, в последней колонке содержится геометрическое (то есть не исправленное на световую задержку) геоцентрическое расстояние кометы, выраженное в астрономических единицах.

5. Возмущения орбит

Расчет движения планет и комет, основанный на решении невозмущенной задачи двух тел, дает эффективный и несложный для понимания метод предсказания положений небесных тел. Простота аналитического описания движения по коническим сечениям также позволяет очень наглядно представить связь между различными элементами орбиты и положением небесного тела.

Однако у этого метода есть и значительные недостатки. Главный из них состоит в том, что он не позволяет учесть влияние больших планет на движение тел Солнечной системы. Конечно, тяготение планет обычно на много порядков слабее солнечной гравитации, однако, действуя на протяжении многих лет, оно может приводить к значительным отклонениям реальной траектории от невозмущенной орбиты. Возмущения особенно значительны для комет и астероидов, проходящих вблизи больших планет. В таких случаях орбита может существенно измениться, что и происходит со многими периодическими кометами. Например, наклонение орбиты кометы Понс-Виннека (7P/1819 L1) в период с 1819 по 1976 годы изменилось с 10° до 22° вследствие того, что ее афелий находится в районе орбиты Юпитера. Другой пример: период обращения кометы Вольфа (14P/1884 S1) после встречи с Юпитером в 1922 году увеличился с 6,8 до 8,3 года.

Хотя влияние планет на орбиты большинства астероидов значительно меньше, чем в приведенных примерах, но и здесь использование простых кеплеровых орбит без учета возмущений не позволяет достичь удовлетворительной точности при долгосрочных расчетах. Все это означает, что для обнаружения и идентификации малых планет при вычислении их эфемерид совершенно необходимо принимать в расчет возмущения. Существует два основных способа, которыми можно для этого воспользоваться. Можно, задав положение и скорость астероида в начальную эпоху, определить затем его траекторию, выполнив численное интегрирование уравнений движения с учетом притяжения Солнца и планет. Другой подход состоит в использовании общей теории возмущений. Отклонения от невозмущенной орбиты представляются в форме тригонометрических рядов, по которым в дальнейшем можно вычислять положения с учетом возмущений на любой момент времени. Однако преимущества этого аналитического метода практически сводятся на нет высокой трудоемкостью составления необходимых рядов, которая не позволяет применять его для тысяч известных астероидов. На практике такие аналитические ряды построены только для больших планет (см. главу 6).

5.1. Уравнение движения

В основе численного метода расчета движения астероидов лежит уравнение движения, которое выражает ускорение объекта как функцию его положения и момента времени. По закону всемирного тяготения на тело массой m в гравитационном поле Солнца действует сила

$$F = ma = GM_{\odot} m \frac{1}{r^2}, \tag{5.1}$$

убывающая обратно пропорционально квадрату расстояния r от Солнца. Напомним еще раз (см. главу 4), что величина

$$GM_{\odot} = k^2 (\text{а. е.})^3 (\text{сут.})^{-2}, \quad \text{где} \quad k = 0{,}01720209895$$

представляет собой произведение гравитационной постоянной и массы Солнца. Учитывая, что сила всегда направлена в сторону Солнца, мы можем представить уравнение (5.1) в векторной форме, домножив его на единичный вектор $-\mathbf{r}/r$

$$\mathbf{a} = -GM_{\odot} \frac{\mathbf{r}}{r^3}. \tag{5.2}$$

Как видим, ускорение не зависит от массы тела m, поскольку сила тяготения сама усиливается пропорционально m.

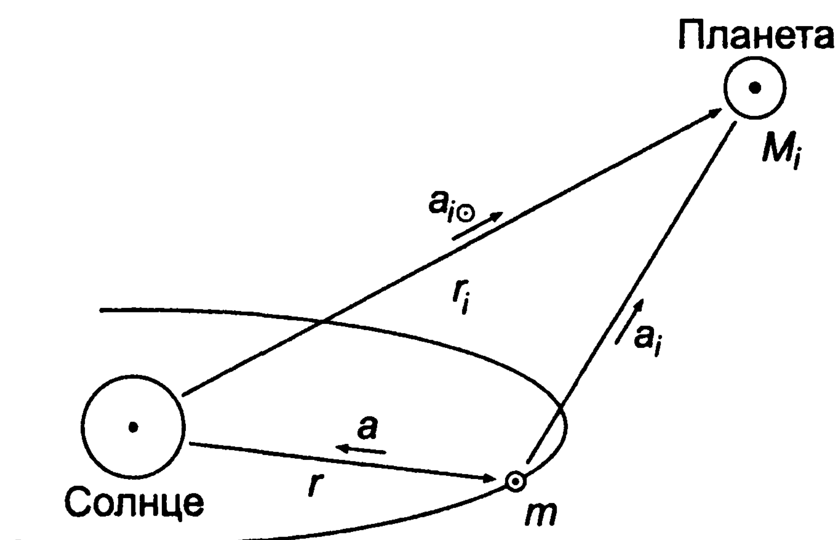


Рис. 5.1. Ускорение тела m под действием Солнца и планет

Аналогично, сила притяжения к планете массой M_i, находящейся в точке $\mathbf{r}_i$ (рис. 5.1), дает вклад в ускорение рассматриваемого тела, равный

$$\boldsymbol{a}_i = -GM_i \frac{\boldsymbol{r} - \boldsymbol{r}_i}{|\boldsymbol{r} - \boldsymbol{r}_i|^3}. \tag{5.3}$$

Надо, однако, еще учесть ускорение, которое испытывает Солнце под влиянием тяготения планеты

$$\boldsymbol{a}_{i\odot} = -GM_i \frac{-\boldsymbol{r}_i}{|-\boldsymbol{r}_i|^3}. \tag{5.4}$$

Вычитая эти величины, получаем вклад планеты в ускорение массы m относительно Солнца

$$\Delta \boldsymbol{a}_i = \boldsymbol{a}_i - \boldsymbol{a}_{i\odot} = -GM_i \left(\frac{\boldsymbol{r} - \boldsymbol{r}_i}{|\boldsymbol{r} - \boldsymbol{r}_i|^3} + \frac{\boldsymbol{r}_i}{|\boldsymbol{r}_i|^3} \right). \quad (5.5)$$

Первый член в этом уравнении называют *прямым пертурбационным ускорением*, а второй — косвенным.

Если учесть влияние всех девяти планет, то получается следующее уравнение для вычисления ускорения кометы или астероида:

$$\boldsymbol{a} = -GM_\odot \frac{\boldsymbol{r}}{r^3} - \sum_{i=1}^{9} GM_i \left(\frac{\boldsymbol{r} - \boldsymbol{r}_i}{|\boldsymbol{r} - \boldsymbol{r}_i|^3} + \frac{\boldsymbol{r}_i}{|\boldsymbol{r}_i|^3} \right). \quad (5.6)$$

В табл. 5.1 приведены соотношения масс Солнца и планет, и из нее видно, что в обычно даже крупнейшие планеты — Юпитер и Сатурн — будут вызывать лишь незначительные возмущения. Если, однако, расстояние до планеты станет меньше 1/10 а. е., а особенно меньше 1/100 а. е., тогда соответствующие компоненты ускорения могут стать того же порядка, что и ускорение, вызванное притяжением Солнца.

Таблица 5.1. Отношение планетных масс к массе Солнца ($M_\odot$) (МАС, 1976)

Планета	Отношение масс
Меркурий	$M_\odot/M_1 = 6023600{,}0$
Венера	$M_\odot/M_2 = 408523{,}5$
Земля/Луна	$M_\odot/M_3 = 328900{,}5$
Марс	$M_\odot/M_4 = 3098710{,}0$
Юпитер	$M_\odot/M_5 = 1047{,}355$
Сатурн	$M_\odot/M_6 = 3498{,}5$
Уран	$M_\odot/M_7 = 22869{,}0$
Нептун	$M_\odot/M_8 = 19314{,}0$
Плутон	$M_\odot/M_9 = 3000000{,}0$

Ускорение $\boldsymbol{a} = (a_x, a_y, a_z)$, действующее на тело, с гелиоцентрическими координатами $\boldsymbol{r} = (x, y, z)$ в момент времени

$$T = \frac{\mathrm{JD} - 2451545}{36525} = \frac{\mathrm{MJD} - 51544{,}5}{36525}$$

вычисляется при помощи функции Accel. В ней предполагается, что нам доступна функция для определения гелиоцентрических положений планет

```
Vec3D Position (PlanetType Planet, double T);
```

использующая тип

```
enum PlanetType { Sun, Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto };
```

Хотя в принципе можно было бы использовать любую систему координат, однако в дальнейшем мы предполагаем, что все координаты отнесены к эпохе наблюдения. Функция Accel затем приводит их к эпохе J2000, в которой выполняется работа с уравнением движения.

```
//-----------------------------------------------------------------------
// Accel. Вычисляет вектор ускорения для малых тел Солнечной системы
//   Mjd       Время в форме модифицированной юлианской даты
//   r         Эклиптические координаты малого тела на эпоху J2000
// <return>:  Эклиптическое ускорение на эпоху J2000 [а.е./сут.**2]
//-----------------------------------------------------------------------
Vec3D Accel (double Mjd, const Vec3D& r)
{
  // Грав. пост. * массы Солнца и планет [а.е.**3/сут.**2]
  static const double  GM[10] = {
                         GM_Sun,                 // Солнце
                         GM_Sun / 6023600.0,     // Меркурий
                         GM_Sun /  408523.5,     // Венера
                         GM_Sun /  328900.5,     // Земля
                         GM_Sun / 3098710.0,     // Марс
                         GM_Sun /    1047.355,   // Юпитер
                         GM_Sun /    3498.5,     // Сатурн
                         GM_Sun /   22869.0,     // Уран
                         GM_Sun /   19314.0,     // Нептун
                         GM_Sun / 3000000.0      // Плутон
                       };
  // Переменные
  int         iPlanet;
  PlanetType  Planet;
  Vec3D       a, r_p, d;
  Mat3D       P;
  double      T, D;

  // Солнечное притяжение
  D = Norm(r);
  a = - GM_Sun * r / (D*D*D);

  // Планетные возмущения
  T = ( Mjd - MJD_J2000 ) / 36525.0;
  P = PrecMatrix_Ecl ( T, T_J2000 );      // Прецессия

  for (iPlanet=Mercury; iPlanet<=Neptune; iPlanet++) {
    Planet = (PlanetType) iPlanet;
    // Положение планеты (эклиптические координаты, эпоха J2000)
//  r_p = P * KepPosition(Planet,T);     // Быстро, но с низкой точностью
    r_p = P * PertPosition(Planet,T);    // Точно, но медленно
    d   = r - r_p;
    // Прямое ускорение
    D = Norm(d);
    a += - GM[iPlanet] * d / (D*D*D);
    // Косвенное ускорение
    D = Norm(r_p);
    a += - GM[iPlanet] * r_p / (D*D*D);
  }
  return a;
}
```

5.2. Планетные координаты

Для того чтобы вычислить ускорение по формуле (5.6) нам нужно знать гелиоцентрические координаты $\boldsymbol{r}$ рассматриваемого тела, а также координаты $\boldsymbol{r}_i$ всех девяти планет — от Меркурия до Плутона. Для высокоточных расчетов можно

использовать заранее вычисленные эфемериды планет. Такие эфемериды предоставляет для научных расчетов, например, Лаборатория реактивного движения НАСА. Они очень точны, но в них используется около 10 000 ежегодно уточняемых коэффициентов, что, конечно, крайне затрудняет их применение.К счастью, в большинстве случаев для учета возмущений кометных и астероидных орбит с точностью до одной секунды дуги достаточно гораздо более простых приблизительных эфемерид. Возмущающие ускорения обычно малы по сравнению с ускорением, вызываемым Солнцем. Поэтому кроме случаев тесных сближений небольшие ошибки в координатах планет приводят к незначительными, второго порядка ошибкам в определении возмущенных координат тела.

Простейший и самый очевидный способ получения координат планет состоит в использовании кеплеровских орбит. Однако точность в этом случае получается невысокой, особенно для внешних планет. В результате ошибка прогноза, например, на десятилетний период, может составить несколько минут дуги. Элементы орбит планет на период с 1995 по 2005 год приведены в табл. 5.2. Их можно применять в функциях `State` и `KepPosition` для вычисления приблизительных положений (и скоростей) планет.

Таблица 5.2. Элементы орбит планет на эпоху 2000, январь 1,5 (JD 2451545,0), отнесенные к эклиптике и равноденствию J2000

Планета	*a* (а. е.)	*e*	*M*	*n*	☊	*i*	ω
Меркурий	0,387099	0,205634	174°794	149472°6738/100 лет	48°331	7°004	77°455
Венера	0,723332	0,006773	50°407	58517°8149/100 лет	76°680	3°394	131°571
Земля	1,000000	0,016709	357°525	35999°3720/100 лет	174°876	0°000	102°940
Марс	1,523692	0,093405	19°387	19140°3023/100 лет	49°557	1°849	336°059
Юпитер	5,204267	0,048775	18°819	3033°6272/100 лет	100°491	1°305	15°558
Сатурн	9,582018	0,055723	320°348	1213°8664/100 лет	113°643	2°485	89°657
Уран	19,229412	0,044406	142°956	426°9282/100 лет	73°989	0°773	170°531
Нептун	30,103658	0,011214	267°765	217°9599/100 лет	131°794	1°768	37°444
Плутон	39,264230	0,244672	15°023	146°3183/100 лет	110°287	17°151	224°050

```
//----------------------------------------------------------------------
// State: Вычисляет положение и скорость по элементам кеплеровой орбиты
//          (используется функциями KepPosition() и KepVelocity())
//   Planet  Идентификатор планеты
//   T       Время в юлианских столетиях с эпохи J2000
//   r       Гелиоцентрическое положение [а.е.]
//           относительно эклиптики и равноденствия на текущую дату
//   v       Гелиоцентрическая скорость [а е./сут ]
//           относительно эклиптики и равноденствия на текущую дату
//----------------------------------------------------------------------
void State (PlanetType Planet, double T, Vec3D& r, Vec3D& v)
{
  const double p = 1.3970;           // Прецессия за сто лет [град.]
  double a, e, M0, O, i, w, n, T0;
  Mat3D  PQR;
  // Элементы орбиты: эклиптика и равноденствие J2000
```

```
  switch ( Planet )
  {
    case Sun:
      r = v = Vec3D(),  // Null vector
      return;
    case Mercury
      a =  0 387099; e = 0 205634; M0 = 174 7947, n = 149472 6738,
      O =  48 331;   i = 7 0048;   w  =  77.4552; T0 = 0.0;
      break;
    case Venus.
      a =  0 723332; e = 0.006773; M0 =  50.4071; n  = 58517 8149;
      O =  76.680;   i = 3 3946;   w  = 131 5718, T0 = 0 0;
      break,
    case Earth:
      a =  1 000000; e = 0.016709; M0 = 357 5256; n  = 35999 3720;
      O = 174.876;   i = 0.0000;   w  = 102.9400, T0 = 0 0,
      break;
    case Mars.
      a =  1 523692; e = 0.093405; M0 =  19.3879; n  = 19140.3023,
      O =  49.557;   i = 1.8496;   w  = 336.0590; T0 = 0 0,
      break;
    case Jupiter:
      a =  5 204267; e = 0.048775; M0 =  18 8185, n  =  3033 6272,
      O = 100 4908;  i = 1.3046;   w  =  15.5576; T0 = 0 0;
      break;
    case Saturn:
      a =  9.582018; e = 0.055723; M0 = 320.3477, n  =  1213 8664;
      O = 113.6427,  i = 2.4852;   w  =  89.6567, T0 = 0 0;
      break,
    case Uranus
      a = 19.229412, e = 0.044406, M0 = 142.9559; n  =   426 9282,
      O =  73.9893,  i = 0.7726,   w  = 170.5310, T0 = 0.0,
      break;
    case Neptune:
      a = 30 103658, e = 0 011214, M0 = 267 7649, n  =   217 9599,
      O = 131.7942,  i = 1 7680;   w  =  37.4435, T0 = 0 0;
      break;
    case Pluto.
      a = 39.264230; e = 0.244672; M0 =  15 0233, n  =   146 3183,
      O = 110.2867;  i = 17.1514;  w =  224 0499, T0 = 0 0;
  }
  // Вычисляем векторы относительно текущих эклиптики и равноденствия
  Ellip ( GM_Sun, Rad*(M0+n*(T-T0)), a, e, r, v ),  // Относительно плоскости орбиты
  PQR = GaussVec ( Rad*(O+p*T), Rad*i, Rad*(w-O) ), // Преобр. к эклиптическим
  r = PQR*r;                                        // Эклиптические
  v = PQR*v,                                        // координаты
}

//-----------------------------------------------------------------------
// KepPosition. Определяет положение планеты по элементам кеплеровой орбиты
//   Planet     Идентификатор планеты
//   T          Время в юлианских столетиях с эпохи J2000
//   <return>   Гелиоцентрическое положение [а. е.]
//              относительно эклиптики и равноденствия на текущую дату
//-----------------------------------------------------------------------
Vec3D KepPosition (PlanetType Planet, double T)
```

```
{
  Vec3D r, v;
  State ( Planet, T, r, v );
  return r;
}
```

Вместо функции KepPosition мы можем, как уже отмечалось выше, использовать в функции Accel функцию PertPosition, которая вычисляет возмущенные координаты планет. PertPosition основана на числовых рядах и описывается в главе 6. Она обеспечивает значительно более точное определение координат, чем KepPosition, особенно для внешних планет. Но за это преимущество приходится платить заметным увеличением времени вычислений.

5.3. Численное интегрирование

Чтобы упростить дальнейшее изложение, объединим радиус-вектор $\boldsymbol{r}$ и вектор скорости $\boldsymbol{v}$ в единый шестимерный вектор

$$\boldsymbol{y}(t)=\left(\frac{\boldsymbol{r}(t)}{\boldsymbol{v}(t)}\right)=\begin{pmatrix}x(t)\\y(t)\\z(t)\\\dot{x}(t)\\\dot{y}(t)\\\dot{z}(t)\end{pmatrix}, \tag{5.7}$$

называемый вектором состояния. Изменение $\dot{\boldsymbol{y}}$ положения и скорости во времени можно представить дифференциальным уравнением

$$\dot{\boldsymbol{y}}=\boldsymbol{f}(t,\boldsymbol{y}), \tag{5.8}$$

где

$$\boldsymbol{f}(t,\boldsymbol{y})=\begin{pmatrix}\boldsymbol{v}(t)\\\boldsymbol{a}(t,\boldsymbol{r})\end{pmatrix} \tag{5.9}$$

функция, объединяющая скорость и ускорение, которые, в свою очередь, являются функциями времени и положения.

С использованием этих обозначений основной принцип процедуры численного интегрирования можно выразить очень просто. Если мы знаем вектор состояния $\boldsymbol{y}_0$ на момент t_0, то можем вычислить приближенное значение вектора состояния на момент $t_1=t_0+h$, пользуясь приближенным равенством

$$\boldsymbol{y}(t_1)\approx\boldsymbol{y}_1=\boldsymbol{y}(t_0)+h\cdot\dot{\boldsymbol{y}}(t_0)=\boldsymbol{y}_0+h\cdot\boldsymbol{f}(t_0,\boldsymbol{y}_0). \tag{5.10}$$

Выполнив это вычисление, мы можем затем по значениям t_1 и $\boldsymbol{y}_1$ получить приближенное значение вектора состояния на момент $t_2=t_0+2h$. Действуя далее аналогичным образом, можно приближенно вычислить положения и скорости для следующих моментов времени: $t_i=t_0+ih$. К сожалению, эта простая

процедура не годится для практического использования, так как дает удовлетворительную точность только при очень малых значениях h, а это означает, что для вычисления орбиты нам понадобится слишком много шагов.

Чтобы получать приемлемые результаты при больших значениях шага, в нашей программе применяется так называемый многошаговый метод, в котором новый вектор состояния вычисляется на основе нескольких предыдущих значений $\boldsymbol{f}$. Для иллюстрации принципа, лежащего в основе этой процедуры, предположим, что у нас уже есть приближенные значения $\boldsymbol{y}_j$ вектора состояния $\boldsymbol{y}(t_j)$ для моментов времени $t_j = t_0 + jh$, где $j = 0, 1, \ldots, i$. Если мы теперь проинтегрируем обе части уравнения

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(t, \boldsymbol{y}) \tag{5.11}$$

по времени t в интервале от t_i до t_{i+1}, то получим эквивалентное уравнение

$$\boldsymbol{y}(t_{i+1}) = \boldsymbol{y}(t_i) + \int_{t_i}^{t_i+h} \boldsymbol{f}(t, \boldsymbol{y}(t))dt. \tag{5.12}$$

В этой форме интеграл нельзя вычислить непосредственно, поскольку он зависит от неизвестного вектора состояния $\boldsymbol{y}(t)$. Чтобы обойти эту проблему, заменим подынтегральное выражение полиномом $\boldsymbol{p}(t)$, интерполирующим значения функции

$$\boldsymbol{f}_j = \boldsymbol{f}(t_j, \boldsymbol{y}_j) \tag{5.13}$$

в более ранних точках t_j, где ее значения по сделанному выше предположению уже известны.

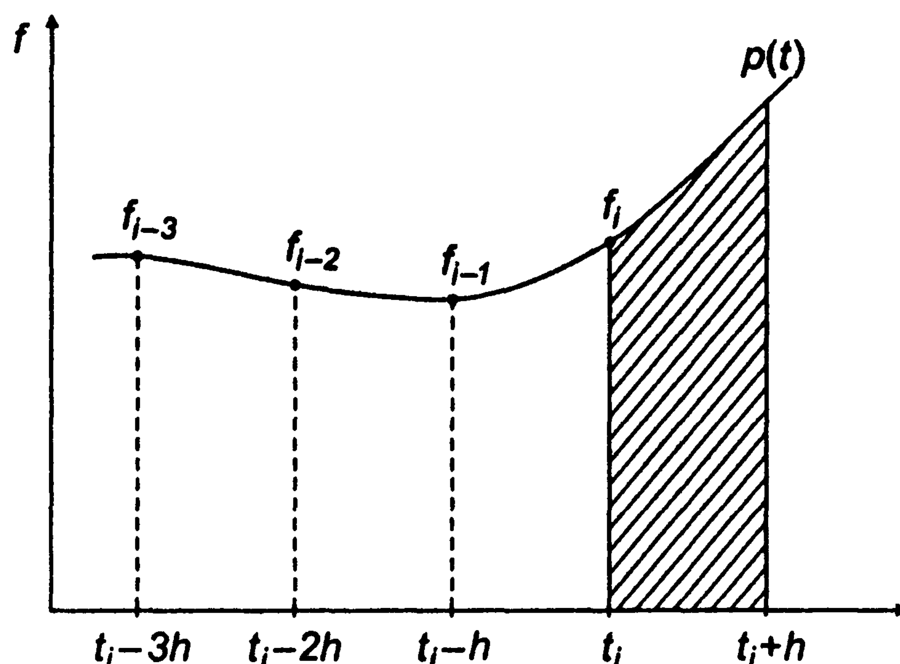


Рис. 5.2. Интерполяция ранних значений функции и интегрирование на следующем интервале времени

В качестве примера на рис. 5.2. показан полином третьего порядка, который определяется четырьмя значениями $\boldsymbol{f}_{i-3}$, $\boldsymbol{f}_{i-2}$, $\boldsymbol{f}_{i-1}$ и $\boldsymbol{f}_i$, соответствующими моментам времени t_{i-3}, t_{i-2}, t_{i-1} и t_i. Если записать этот полином в форме

$$\boldsymbol{p}(t) = \boldsymbol{a}_0 + \boldsymbol{a}_1\sigma + \boldsymbol{a}_2\sigma^2 + \boldsymbol{a}_3\sigma^3, \quad \text{где} \quad \sigma = \frac{1}{h}(t - t_i), \tag{5.14}$$

то коэффициенты будут вычисляться по следующим формулам:

$$\begin{aligned} \boldsymbol{a}_0 &= (6\boldsymbol{f}_i)/6, \\ \boldsymbol{a}_1 &= (-2\boldsymbol{f}_{i-3} + 9\boldsymbol{f}_{i-2} - 18\boldsymbol{f}_{i-1} + 11\boldsymbol{f}_i)/6, \\ \boldsymbol{a}_2 &= (-3\boldsymbol{f}_{i-3} + 12\boldsymbol{f}_{i-2} - 15\boldsymbol{f}_{i-1} + 6\boldsymbol{f}_i)/6, \\ \boldsymbol{a}_3 &= (-1\boldsymbol{f}_{i-3} + 3\boldsymbol{f}_{i-2} - 3\boldsymbol{f}_{i-1} + 1\boldsymbol{f}_i)/6\,. \end{aligned} \tag{5.15}$$

В результате мы получаем уравнение Адамса—Башфорта четвертого порядка

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + \int_{t_i}^{t_i+h} \boldsymbol{p}(t)dt = \boldsymbol{y}_i + \frac{h}{24}(-9\boldsymbol{f}_{i-3} + 37\boldsymbol{f}_{i-2} - 59\boldsymbol{f}_{i-1} + 55\boldsymbol{f}_i), \tag{5.16}$$

которое дает хорошее приближение для вектора состояния при достаточно больших значениях шага. Повторение этой процедуры позволяет нам получить значения для последующих моментов времени $t_i + jh$.

В общем случае, чем больше членов используется в интерполяционном полиноме (то есть чем выше его степень), тем выше точность вычислений. Однако при больших значениях шага могут возникать неустойчивости и поэтому степень полинома и величину шага следует выбирать в зависимости от необходимой точности.

Метод DE, используемый нами для вычисления возмущенных орбит малых планет, основан на многошаговой процедуре Адамса. Однако, в отличие от приведенного выше упрощенного описания, где использовались фиксированные значения степени полинома и шага интегрирования, метод DE обладает рядом полезных на практике особенностей:

- Количество используемых предшествующих точек, а значит, и степень полинома не задается, а самостоятельно выбирается программой DE. Когда требуется предельно высокая точность, DE использует полиномы двенадцатого порядка.
- Величина шага является переменной. Ее выбор зависит от изменения вектора состояния во времени. Это особенно важно при работе с сильно вытянутыми орбитами, для которых в перигелии требуется использовать значительно меньший шаг, чем в афелии.
- В отличие от описанной нами упрощенной процедуры с фиксированными порядком и шагом, программа DE не требует задания нескольких начальных точек $\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots$ Вместо этого DE начинает вычисление, используя полином первого порядка с малой величиной шага, а затем увеличивает порядок полинома и величину шага, пока они не достигнут оптимальных значений. Иначе говоря, DE — самостартующая процедура.
- Значение вектора состояния $\boldsymbol{y}(t)$ можно получить для любого момента времени. Для этого значения, полученные в ходе вычислений, автоматически интерполируются. Тем самым размер шага не влияет на выбор точек, для которых будут получены результаты.

Метод DE был разработан Л. Ф. Шемпайном (L. F. Shampine) и М. К. Гордоном (M. K. Gordon)и реализован в виде программы на Фортране. Ввиду ограниченно-

сти места мы не будем приводить здесь весьма обширный листинг этой программы и обсуждать детали ее реализации. Заинтересованным читателям мы рекомендуем обратиться к книге *Computer Solution of Ordinary Differential Equations*, где детально описаны особенности этого кода.

Для наших целей мы перевели программу DE на язык C++ и адаптировали к его особенностям. Большие преимущества мы получили, определив специальный класс SolverDE, который почти полностью освобождает пользователя от заботы о таких технических подробностях, как выделение необходимого рабочего объема памяти.

```
// Коды состояний для интегратора SolverDE
enum DE_STATE {
  DE_INIT                    = 1,    // Начальный шаг
  DE_DONE                    = 2,    // Число выполненных шагов интегрирования
  DE_ACCURACY_NOT_ACHIEVED = 3,    // Затребована слишком высокая точность
  DE_TOO_MANY_STEPS          = 4,    // Требуется слишком много шагов
  DE_STIFF                   = 5,    // Вероятно, дифференциальное уравнение жесткое
  DE_INVALID_PARAMS          = 6     // Неверный ввод
};

//
// Класс-интегратор SolverDE
//
class SolverDE
{
  public:
    // Конструктор, деструктор
    SolverDE (DEfunct F, int Neqn);
    ~SolverDE ();
    // Интегрирование
    void Integ (
      double y[],        // Вектор-решение: обновляется при выводе
      double& t,         // Значение независимой переменной, обновляется при выводе
      double tout,       // Верхняя граница интегрирования
      double& relerr,    // Желательная относительная точность решения: обновленная
      double& abserr,    // Желательная абсолютная точность решения, обновленная
      DE_STATE& State,   // Код состояния
      bool PermitTOUT = true
    );
  private:
    // Члены-данные
    DEfunct  f;
    int      neqn;
    double   *yy,*wt,*p,*yp,*ypout;
    double   **phi;
    double   alpha[13],beta[13],v[13],w[13],psi[13];
    double   sig[14],g[14];
    double   x,h,hold,told,delsgn;
    int      ns,k,kold;
    bool     OldPermit, phase1,start,nornd;
    // Интерполяция
    void Intrp ( double x, const double y[], double xout, double yout[], double ypout[] );
    // Элементарный шаг интегрирования
    void Step (double& x, double y[], double& eps, bool& crash);
};
```

5.4. Оскулирующие элементы

В случае невозмущенной кеплеровой орбиты движение тела вокруг Солнца задается шестью элементами: большой полуосью a, эксцентриситетом e, наклонением i, долготой восходящего узла Ω, долготой перигелия ω и средней аномалией M. По этим элементам орбиты можно вычислить положение и скорость тела в любой заданный момент времени (см. главу 4).

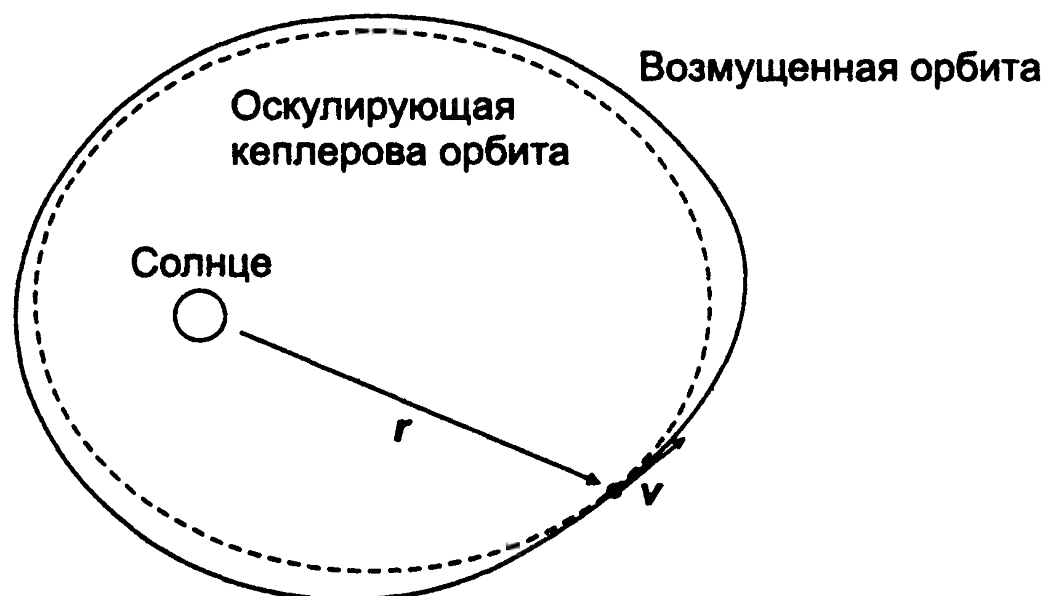


Рис. 5.3. Оскулирующие элементы позволяют задать возмущенную орбиту

Геометрический смысл элементов орбиты настолько нагляден, что их полезно использовать и для описания возмущенных орбит астероидов и комет. Мы будем опираться на тот факт, что элементы орбиты и вектор состояния взаимно однозначно определяют друг друга. Если в определенный момент времени t мы знаем положение $\boldsymbol{r} = (x, y, z)$ и скорость $\dot{\boldsymbol{r}} = (\dot{x}, \dot{y}, \dot{z})$ небесного тела, то существует единственный такой набор элементов орбиты $(a, e, i, \Omega, \omega, M)$, подставив который в уравнение Кеплера для невозмущенной орбиты мы получим тот же вектор ($\boldsymbol{r}$, ν) состояния. Элементы орбиты для возмущенного движения можно определить аналогично, но они, конечно, уже не будут оставаться постоянными. В зависимости от характера и величины возмущений они будут испытывать флуктуации во времени. Такие элементы орбиты, определенные по вектору состояния обычно называют *оскулирующими*. Кеплерова орбита, задаваемая оскулирующими элементами, касается истинной орбиты в точке, соответствующей выбранной эпохе, и очень близка к ней на протяжении определенного периода времени (рис. 5.3).

В случае невозмущенного движения плоскость оскулирующей орбиты определяется в любой момент времени векторами положения и скорости (см. рис. 4.1). Это означает, что векторное произведение $\boldsymbol{r}(t)$ и $\dot{\boldsymbol{r}}(t)$

$$\boldsymbol{h} = \boldsymbol{r} \times \dot{\boldsymbol{r}} = \begin{pmatrix} y\dot{z} - z\dot{y} \\ z\dot{x} - x\dot{z} \\ x\dot{y} - y\dot{x} \end{pmatrix} \tag{5.17}$$

определяет вектор $\boldsymbol{h}(t)$, перпендикулярный плоскости орбиты и параллельный гауссову вектору орбиты:

$$\boldsymbol{R} = \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = \begin{pmatrix} +\sin i \, \sin \Omega \\ -\sin i \, \cos \Omega \\ +\cos i \end{pmatrix} = \begin{pmatrix} h_x/h \\ h_y/h \\ h_z/h \end{pmatrix}. \tag{5.18}$$

Отсюда мы можем получить наклонение i оскулирующей орбиты, а также долготу ее восходящего узла Ω:

$$i = \operatorname{tg}^{-1}\left(\frac{\sqrt{h_x^2 + h_y^2}}{h_z}\right), \qquad \Omega = \operatorname{tg}^{-1}\left(\frac{h_x}{-h_y}\right). \tag{5.19}$$

Решая уравнение (4.20) относительно u, получаем дополнительное выражение для аргумента долготы

$$u = \nu + \omega = \operatorname{tg}^{-1}\left(\frac{z\,h}{-x\,h_y + y\,h_x}\right), \tag{5.20}$$

значение которого понадобится нам в дальнейшем для вычисления аргумента перигелия.

Из уравнений (4.5) и (4.10) видно, что большая полуось орбиты зависит только от расстояния r тела от Солнца и абсолютной величины его скорости $v = |\dot{\boldsymbol{r}}|$

$$a = \left(\frac{2}{r} - \frac{v^2}{GM_{\odot}}\right)^{-1}. \tag{5.21}$$

Если скорость тела относительно Солнца меньше скорости освобождения $v_{\infty} = \sqrt{2GM_{\odot}/r}$, то орбита будет эллиптической, и по формуле (5.21) мы получим положительное значение большой полуоси. В этом случае мы можем записать уравнения (4.5) и (4.10) в виде

$$\begin{aligned} e\cos(E) &= 1 - r/a, \\ e\sin(E) &= (x\dot{x} + y\dot{y} + z\dot{z})/\sqrt{GM_{\odot}a}, \end{aligned} \tag{5.22}$$

что дает нам возможность определить эксцентриситет e и эксцентрическую аномалию E. Из уравнения Кеплера (4.7) мы также можем получить значение средней аномалии

$$M = E - \frac{180^\circ}{\pi} e\sin(E) \tag{5.23}$$

в момент времени t. Наконец, по истинной аномалии

$$\nu = \operatorname{tg}^{-1} \frac{\sqrt{1-e^2}\,\sin(E)}{\cos(E) - e}, \tag{5.24}$$

соответствующей значению E, мы получаем шестой элемент орбиты — аргумент перигелия

$$\omega = \nu - u. \tag{5.25}$$

Для элементов параболических и гиперболических орбит существуют аналогичные выражения, однако при расчете орбит астероидов и периодических комет они нам не понадобятся, и поэтому приводимая ниже функция Elements ограничивается случаем эллиптических орбит.

```
//------------------------------------------------------------------------
// Elements· вычисляет элементы эллиптической орбиты по положению тела
//           и вектору его скорости
//   GM      Произведение грав. постоянной и центральной массы [а е ^3 * сут ^-2]
//   r       Прямоугольные гелиоцентрические эклиптические координаты [а.е ]
//   v       Гелиоцентрический эклиптический вектор скорости [а е /сут.]
//   a       Большая полуось орбиты [а е ]
//   e       Эксцентриситет орбиты
//   i       Наклонение орбиты к плоскости эклиптики [рад]
//   Omega   Долгота восходящего узла орбиты [рад]
//   omega   Аргумент перигелия [рад]
//   M       Средняя аномалия [рад]
//------------------------------------------------------------------------
void Elements ( double GM, const Vec3D& r, const Vec3D& v,
                double& a, double& e, double& i,
                double& Omega, double& omega, double& M )
{
  // Variables
  Vec3D  h;
  double H, u, R, v2,
  double eCosE, eSinE, e2, E, nu;

  h = Cross(r,v);                                // Секторная скорость
  H = Norm(h);
  Omega = atan2(h[x], -h[y]),                    // Долгота восх  узла
  i     = atan2(sqrt(h[x]*h[x]+h[y]*h[y]), h[z]),  // Наклонение
  u     = atan2(r[z]*H, -r[x]*h[y]+r[y]*h[x]);   // Аргумент перигелия

  R  = Norm(r);                                  // Расстояние
  v2 = Dot(v, v);                                // Квадрат скорости
  a = 1 0 / (2.0/R-v2/GM);                       // Большая полуось

  eCosE = 1.0-R/a,                               // e*cos(E)
  eSinE = Dot(r, v)/sqrt(GM*a);                  // e*sin(E)
  e2 = eCosE*eCosE +eSinE*eSinE,
  e  = sqrt(e2),                                 // Эксцентриситет
  E  = atan2(eSinE,eCosE),                       // Эксцентрическая аномалия

  M  = E - eSinE,                                // Средняя аномалия
  nu = atan2(sqrt(1.0-e2)*eSinE, eCosE-e2);      // Истинная аномалия
  omega = u - nu;                                // Аргумент перигелия

  if (Omega<0.0) Omega += 2.0*pi,
  if (omega<0.0) omega += 2.0*pi;
  if (M    <0 0) M     += 2 0*pi;
};
```

5.5. Программа Numint

Программа Numint может использоваться для точного расчета орбит астероидов и периодических комет. Ее структура очень похожа на программу Comet, которую

мы обсуждали в главе 4. Однако использование процедуры численного интегрирования позволяет учесть в расчетах наряду с солнечным притяжением возмущения со стороны больших планет. Благодаря этому на длительных интервалах времени удается получать значительно более точные результаты. Расплачиваться за это приходится значительным увеличением времени расчета, которое растет пропорционально величине периода, для которого требуется получить эфемериды. Интегрирование орбиты требует намного больше времени, чем расчет положений по элементам кеплеровой орбиты в программе Comet.

При запуске программы элементы орбиты и соответствующая им эпоха считываются из файла данных Numint.dat и для проверки отображаются на экране. Затем пользователь вводит период времени и шаг эфемериды, а также эпоху, для которой она вычисляется. Данные приводятся к фиксированной эпохе J2000, затем вычисляются положение и скорость астероида, после чего выполняется численное интегрирование орбиты от эпохи, на которую заданы элементы до начального момента эфемериды. Если вы используете элементы орбиты, отнесенные к эпохе на несколько лет раньше той, для которой вычисляете эфемериду, то эта процедура может занять десятки секунд.

Получив, таким образом, вектор состояния, программа Numint затем вычисляет соответствующие оскулирующие элементы и выводит их на экран. По величине изменения элементов орбиты сравнительно с исходными значениями можно оценить масштаб возмущений, вызванных большими планетами. Сохранив значения этих элементов орбиты, можно впоследствии использовать их для того, чтобы при повторном расчете избежать длительных вычислений до выхода на начальный момент эфемериды.

Наконец, завершив все приготовления, программа вычисляет эфемериду астероида для заданного периода и с указанным шагом по времени. Как и в программе Comet, учитывается световая задержка, так что полученные астрометрические координаты можно непосредственно сравнивать со звездными картами и каталогами. Положение Земли относительно Солнца вычисляется с использованием функции SunPos — вновь так же, как в Comet,— что позволяет избежать ошибок при выполнении преобразования координат астероида из гелиоцентрической системы в геоцентрическую.

```
//------------------------------------------------------------------------
// Файл·       Numint cpp
// Назначение· Численное интегрирование возмущенной орбиты астероида
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//------------------------------------------------------------------------
#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Const.h"
#include "APC_DE.h"
#include "APC_IO.h"
#include "APC_Kepler.h"
#include "APC_Math.h"
```

```
#include "APC_Planets.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

using namespace std;

//------------------------------------------------------------------------------
// WriteElm:  Вывод на экран элементов орбиты
//   MjdEpoch Эпоха в форме модифицированной юлианской даты
//   a        Большая полуось орбиты [а.е.]
//   e        Эксцентриситет орбиты
//   i        Наклонение орбиты к эклиптике [рад]
//   Omega    Долгота восходящего узла орбиты [рад]
//   omega    Аргумент перигелия [рад]
//   M        Средняя аномалия [рад]
//   T_eqx0   Эпоха в юлианских столетиях, считая от J2000
//------------------------------------------------------------------------------
void WriteElm ( double MjdEpoch, double a, double e, double i,
                double Omega, double omega, double M, double T_eqx0 )
{
  cout << "  Epoch (y m d)            "
       << "  " << setprecision(2) << DateTime(MjdEpoch,DDd) << endl;
  cout << "  Semi-major axis (a)      "
       << setprecision(7) << setw(14) << a << " AU" << endl;
  cout << "  Eccentricity (e)         "
       << setprecision(7) << setw(14) << e << endl;
  cout << "  Inclination (i)          "
       << setprecision(5) << setw(12) << i*Deg << " deg" << endl;
  cout << "  Long. of ascending node "
       << setprecision(5) << setw(12) << Omega*Deg << " deg" << endl;
  cout << "  Argument of perihelion  "
       << setprecision(5) << setw(12) << omega*Deg << " deg" << endl;
  cout << "  Mean anomaly (M)         "
       << setprecision(5) << setw(12) << M*Deg     << " deg" << endl;
  cout << "  Equinox                  "
       << setprecision(2) << setw(9)  << 2000.0+100*T_eqx0 << endl << endl;
};

//------------------------------------------------------------------------------
// GetElm: Чтение элементов орбиты из входного файла
//   Filename  Имя входного файла с элементами орбиты
//   MjdEpoch Эпоха в форме модифицированной юлианской даты
//   a        Большая полуось орбиты [а.е.]
//   e        Эксцентриситет орбиты
//   M        Средняя аномалия [рад]
//   PQR      Матрица преобразования от плоскости орбиты к эклиптике
//   T_eqx0   Эпоха в юлианских столетиях, считая от J2000
//------------------------------------------------------------------------------
void GetElm ( char* Filename,
              double& MjdEpoch,
              double& a, double& e, double& M, Mat3D& PQR, double& T_eqx0 )
{
  int      year, month;
  double   Omega, i, omega;
```

```
  double    Year, day;
  ifstream  inp;
  // Чтение элементов
  inp.open(Filename);
  inp >> year >> month >> day; inp.ignore(81,'\n');
  inp >> a; inp.ignore(81,'\n');
  inp >> e; inp.ignore(81,'\n');
  inp >> i; inp.ignore(81,'\n');
  inp >> Omega; inp.ignore(81,'\n');
  inp >> omega; inp.ignore(81,'\n');
  inp >> M; inp.ignore(81,'\n');
  inp >> Year; inp.ignore(81,'\n');
  inp.close();
  // Вычисление производных данных
  i*=Rad;  Omega*=Rad;  omega*=Rad;  M*=Rad;
  MjdEpoch = Mjd(year,month,int(day))+(day-int(day));
  T_eqx0   = (Year-2000.0)/100.0;
  PQR      = GaussVec(Omega,i,omega);
  // Распечатка элементов
  cout << " Orbital elements from file " << Filename << ":" << endl << endl;
  WriteElm ( MjdEpoch, a,e,i,Omega,omega,M, T_eqx0 );
}

//------------------------------------------------------------------------------
// GetEph: Предлагает пользователю ввести начальную и конечную даты эфемериды,
//         размер шага таблицы и требуемую эпоху
//   MjdStart  Начальная дата эфемериды в форме модифицированной юлианской даты
//   Step      Размер шага эфемериды в сутках
//   MjdEnd    Конечная дата эфемериды в форме модифицированной юлианской даты
//   T_eqx     Требуемая эпоха, в юлианских столетиях от J2000
//------------------------------------------------------------------------------
void GetEph (double& MjdStart, double& Step, double& MjdEnd, double& T_eqx)
{
  int    year, month, day;
  double hour, Year;
  cout << " Begin and end of the ephemeris: " << endl;
  cout << "  First date (yyyy mm dd hh.hhh)          ... ";
  cin >> year >> month >> day >> hour; cin.ignore(81,'\n');
  MjdStart = Mjd(year,month,day) + hour/24.0 ;
  cout << "  Final date (yyyy mm dd hh.hhh)          ... ";
  cin >> year >> month >> day >> hour; cin.ignore(81,'\n');
  MjdEnd = Mjd(year,month,day) + hour/24.0 ;
  cout << "  Step size (dd hh.hh)                    ... ";
  cin >> day >> hour; cin.ignore(81,'\n');
  Step  = day + hour/24.0;
  cout << endl
       << " Desired equinox of the ephemeris (yyyy.y) ... ";
  cin >> Year; cin.ignore(81,'\n');
  T_eqx = (Year-2000.0) /100.0;
}

//------------------------------------------------------------------------------
//
// Главная программа
//
//------------------------------------------------------------------------------
void main(int argc, char* argv[])
```

```
{
  // Константы
  const pol_index R    = pol_index(2).   // Индекс для доступа к длине вектора
  const int       Neqn = 6:              // Число дифференциальных уравнений
  const double    eps  = 1.0e-10.        // Относительная точность

  // Переменные
  int        n_line:
  double     MjdStart. Step. MjdEnd. T_eqx.
  double     MjdEpoch. a. e. M. T_eqx0:
  double     Omega. i. omega.
  Mat3D      PQR. P:
  double     Mjd. MjdStep. T:
  SolverDE   Orbit(F. Neqn):
  double     Y[Neqn+1]:
  double     relerr.abserr:
  DE_STATE   State:
  Vec3D      r. v.
  Vec3D      R_Sun. r_helioc. v_helioc. r_geoc. r_equ:
  double     dist. fac.
  char       InputFile[APC_MaxFilename] = "".
  char       OutputFile[APC_MaxFilename] = "":
  bool       FoundInputfile = false:
  bool       FoundOutputfile = false.
  ofstream   OutFile:

  // Заголовок
  cout << endl
       << "  NUMINT· numerical integration of perturbed minor planet orbits"
       << endl
       << "          (c) 1999 Oliver Montenbruck. Thomas Pfleger         "
       << endl << endl:

  // Находим входной и выходной файлы
  GetFilenames( argc. argv. "Numint dat". InputFile. FoundInputfile.
                OutputFile. FoundOutputfile ).
  // Завершаем работу. если не найден входной файл
  if (!FoundInputfile) {
    cerr << " Terminating program." << endl:
    exit(-1).
  }

// Считываем элементы орбиты из файла
// и запрашиваем у пользователя параметры эфемериды
  GetElm ( InputFile. MjdEpoch. a. e. M. PQR. T_eqx0 ).
  GetEph ( MjdStart. Step. MjdEnd. T_eqx ).

  // Перенаправляем вывод в случае создания выходного файла
  if (FoundOutputfile) {
    OutFile.open(OutputFile).
    if (OutFile is_open())
      cout = OutFile:
  }

  P = PrecMatrix_Ecl(T_J2000.T_eqx):

  // Начальный вектор состояния (эпоха J2000)
```

```
PQR = PrecMatrix_Ecl(T_eqx0,T_J2000) * PQR;
Ellip ( GM_Sun, M,a,e, r,v );
r = PQR*r;
v = PQR*v;
Y[0]=0.0;  // Не используется
Y[1]=r[x]; Y[2]=r[y]; Y[3]=r[z];
Y[4]=v[x]; Y[5]=v[y]; Y[6]=v[z];
Mjd = MjdEpoch;

// Начало интегрирования: прогоняем вектор состояния от исходной эпохи
// до момента начала эфемериды
State = DE_INIT;
relerr = eps;  abserr = 0.0;
do {
  Orbit.Integ(Y, Mjd, MjdStart, relerr, abserr, State);
  if ( State==DE_INVALID_PARAMS ) {
    cerr << "Exit (invalid parameters)" << endl;
    exit(1);
  }
}
while ( State > DE_DONE );

// Элементы орбиты на момент начала эфемериды (эпоха T_eqx)
r = P*Vec3D(Y[1],Y[2],Y[3]);
v = P*Vec3D(Y[4],Y[5],Y[6]);
Elements (GM_Sun, r, v, a, e, i, Omega, omega, M);

// Распечатка элементов орбиты
cout << endl << endl
     << " Orbital elements at start epoch:" << endl
     << endl;
WriteElm ( Mjd, a,e,i,Omega,omega,M, T_eqx );

// Заголовок эфемериды
cout << endl << endl
     << "    Date    ET    Sun      l       b      r"
     << "       RA          Dec        Distance " << endl
     << setw(43) << " " << "   h  m  s      o  '  \"    (AU) " << endl;

// Вычисляем эфемериду
n_line = 0;
MjdStep = MjdStart;

// Цикл по времени
while ( MjdStep < MjdEnd + Step/2 ) {
  // Интегрируем орбиту до MjdStep
  do {
    Orbit.Integ(Y, Mjd, MjdStep, relerr, abserr, State);
    if (State==DE_INVALID_PARAMS) {
      cerr << "Exit (invalid parameters)" << endl;
      exit(1);
    }
  }
  while ( State > DE_DONE );
  // Гелиоцентрические эклиптические координаты, эпоха T_eqx
  r_helioc = P*Vec3D(Y[1],Y[2],Y[3]);
  v_helioc = P*Vec3D(Y[4],Y[5],Y[6]);
```

```
    // Геоцентрические эклиптические координаты Солнца, эпоха T_eqx
    T = ( Mjd - MJD_J2000 ) / 36525.0;
    R_Sun = PrecMatrix_Ecl(T,T_eqx) * SunPos(T);
    // Геометрические геоцентрические координаты
    r_geoc = r_helioc + R_Sun;
    // Поправка за световую задержку (в первом порядке)
    dist = Norm(r_geoc);
    fac  = 0.00578*dist;
    r_geoc = r_geoc - fac*v_helioc;
    // Экваториальные координаты
    r_equ = Ecl2EquMatrix(T_eqx) * r_geoc;
    // Вывод
    cout << DateTime(Mjd,HHh)
         << fixed << setprecision(1)
         << setw(7) << Deg*R_Sun[phi]
         << setw(7) << Deg*r_helioc[phi]
         << setw(6) << Deg*r_helioc[theta]
         << setprecision(3) << setw(7) << r_helioc[R]
         << setprecision(1) << setw(12) << Angle(Deg*r_equ[phi]/15.0,DMMSSs)
         << "  " << showpos << setw(9) << Angle(Deg*r_equ[theta],DMMSS)
         << noshowpos << setprecision(6) << setw(11) << dist
         << endl;
    ++n_line;
    if ( (n_line % 5) ==0 ) cout << endl; // Вставляет пустую строку каждые пять строк
    MjdStep += Step;  // Next time step
  } // Конец цикла по времени
  if (OutFile.is_open()) OutFile.close();
}
```

В качестве примера использования данной программы вычислим эфемериду малой планеты № 1, Цереры, вблизи ее противостояния в 1992 году. При этом мы будем использовать оскулирующие элементы на 1983 год (табл. 5.3). Первая строка файла Numint.dat содержит момент, на который определены приводимые в следующих строках шесть элементов орбиты *a*, *e*, *i*, ☊, ω, *M*. В последней строке приводится эпоха, к которой отнесены элементы орбиты.

```
1983 09 23.0    ! Эпоха (год месяц день.доля_дня) для элементов астероида 1 (Церера)
   2.7657991    ! Большая полуось в а.е.
   0.0785650    ! Эксцентриситет e
  10.60646      ! Наклонение i (град.)
  80.05225      ! Долгота восходящего узла (град )
  73.07274      ! Аргумент перигелия (град.)
 174.19016      ! Средняя аномалия (град.)
1950.0          ! Равноденствие элементов орбиты
```

Соответствующие данные для других астероидов ежегодно публикуются в сборниках «Эфемериды малых планет», издаваемом Институтом прикладной астрономии в Санкт-Петербурге, и в «Minor Planet Circular», выпускаемом Смитсониановской астрофизической обсерваторией, а также в ряде других ежегодников.

Данные о периоде, для которого вычисляется эфемерида, шаге по времени и эпохе вводятся аналогично тому, как это делается в программе Comet. Как обычно, ввод пользователя выделен курсивом.

Таблица 5.3. Оскулирующие элементы астероида Церера

Эпоха		1983 сентябрь 23,0 ET	1992 январь 27,0 ET
a		2,7657991 а. е.	2,7674385 а. е.
e		0,0785650	0,0765551
i	(1950)	10°,60646	10°,59950
	(2000)	10°,60695	10°,59999
☊	(1950)	80°,05225	80°,01288
	(2000)	80°,71588	80°,67649
ω	(1950)	73°,07274	71°,07929
	(2000)	73°,10812	71°,11469
M		174°,19016	141°,46349

```
 NUMINT: numerical integration of perturbed minor planet orbits
         (c) 1999 Oliver Montenbruck, Thomas Pfleger

Using default input file Numint.dat
Orbital elements from file Numint.dat:

 Epoch (y m d)              1983/09/23.00
 Semi.major axis (a)           2.7657991 AU
 Eccentricity (e)              0.0785650
 Inclination (i)              10.60646 deg
 Long. of ascending node      80.05225 deg
 Argument of perihelion       73.07274 deg
 Mean anomaly (M)            174.19016 deg
 Equinox                    1950.00

Begin and end of the ephemeris:
 First date (yyyy mm dd hh.hhh)          ... 1992 06 27 00.0
 Final date (yyyy mm dd hh.hhh)          ... 1992 07 25 00.0
 Step size (dd hh.hh)                    ... 2 00.0

Desired equinox of the ephemeris (yyyy.y) ... 2000.0
Orbital elements at start epoch:

 Epoch (y m d)              1992/06/27.00
 Semi.major axis (a)           2.7674372 AU
 Eccentricity (e)              0.0765610
 Inclination (i)              10.59999 deg
 Long. of ascending node      80.67653 deg
 Argument of perihelion       71.11641 deg
 Mean anomaly (M)            141.46191 deg
 Equinox                    2000.00

   Date     ET    Sun     l      b      r      RA          Dec       Distance
                                             h  m  s     o  '  "      (AU)
1992/06/27 00.0   95.7  297.9  -6.5  2.939  20 55 54.5  -27 00 46   2.042458
1992/06/29 00.0   97.6  298.2  -6.5  2.940  20 54 53.4  -27 13 47   2.029192
1992/07/01 00.0   99.5  298.6  -6.6  2.941  20 53 46.7  -27 26 56   2.016828
1992/07/03 00.0  101.4  299.0  -6.6  2.942  20 52 34.7  -27 40 13   2.005397
1992/07/05 00.0  103.3  299.4  -6.7  2.942  20 51 17.5  -27 53 32   1.994924

1992/07/07 00.0  105.2  299.7  -6.7  2.943  20 49 55.6  -28 06 53   1.985432
1992/07/09 00.0  107.1  300.1  -6.8  2.944  20 48 29.0  -28 20 11   1.976941
1992/07/11 00.0  109.0  300.5  -6.8  2.945  20 46 58.3  -28 33 24   1.969468
1992/07/13 00.0  111.0  300.9  -6.9  2.946  20 45 23.6  -28 46 28   1.963030
1992/07/15 00.0  112.9  301.2  -6.9  2.946  20 43 45.4  -28 59 22   1.957641
```

```
1992/07/17 00.0  114.8  301.6  -7.0  2.947  20 42 03.9  -29 12 03   1.953316
1992/07/19 00.0  116.7  302.0  -7.0  2.948  20 40 19.7  -29 24 26   1.950066
1992/07/21 00.0  118.6  302.4  -7 1  2 949  20 38 33.1  -29 36 31   1.947903
1992/07/23 00.0  120.5  302.7  -7 1  2.950  20 36 44.6  -29 48 14   1 946836
1992/07/25 00 0  122.4  303.1  -7.2  2.950  20 34 54.5  -29 59 32   1.946871
```

В приведенном примере для определения точных положений планет используется функция PertPosition. Эти положения затем используются для вычисления планетных возмущений в функции Accel. Сразу после ввода данных программе требуется некоторое время для того, чтобы выполнить интегрирование орбиты на протяжении девяти лет и получить оскулирующие элементы на 27 января 1992 года. Собственно расчет эфемериды на заданный интервал времени занимает доли секунды.

Сравнение с ежегодником показывает, что нам удалось достичь точности около 1″, то есть примерно такой же, как у солнечных координат в процедуре перевода гелиоцентрических координат в геоцентрические.

Если выполнять вычисления без учета возмущений, вызываемых планетами (это можно сделать, например, закомментировав соответствующие строки в функции Accel), то будет получена следующая эфемерида:

```
    Date     ET     Sun     l       b    r        RA          Dec     Distance
                                                h  m  s      o  '  "    (AU)
1992/06/27 00 0    95.7 298 6  -6 6 2.938  21 00 35.9  -26 49 55  2.049522
1992/06/29 00 0    97.6 299 0  -6 6 2.939  20 59 37 9  -27 02 56  2.035891
1992/07/23 00 0   120 5 303 5  -7 2 2 950  20 41 55.4  -29 39 16  1.948277
1992/07/25 00 0   122.4 303.9  -7.3 2.951  20 40 06.3  -29 50 51  1.947831
```

Видно, что возмущениям наиболее подвержена (0°7–0°8) гелиоцентрическая долгота астероида. Возмущение в широте не так велико, но тоже довольно значительно — 0°1. Ввиду небольшого расстояния от Земли прямое восхождение, определенное по невозмущенной эфемериде, будет отличаться от действительного значения примерно на 5 минут времени или на 1°3.

5.6. База данных элементов орбит астероидов

На сегодня открыто более 50 000 астероидов. Их орбиты регулярно уточняются по результатам новых наблюдений. За присвоение обозначений и каталогизацию астероидов отвечает Центр малых планет при Смитсониановской астрофизической обсерватории в Массачусетсе. Кроме того, Лоувелловская обсерватория предоставляет свободный доступ к каталогу элементов орбит и другой информации, которую готовит и постоянно обновляет Тед Боувелл. С разрешения автора этот каталог вместе с полным описанием включен в состав прилагаемого CD-ROM[1]. Текущая версия файла astorb.dat (на январь 2000 года) содержит более 58 000 записей фиксированной длины по 267 символов. С этим файлом мож-

[1] Обновленную версию AOE Database можно получить в Интернете по адресу **ftp://ftp.lowell.edu/pub/elgb/astorb.dat** или **.../astorb.gz+**.

но работать в любом текстовом редакторе. Размер файла составляет около 15 мегабайт. Смысл большинства важных полей в этом файле поясняется в табл. 5.4.

Таблица 5.4. Структура базы данных элементов орбит астероидов (краткая сводка)

Позиции	Описание
1–5	Номер астероида
7–24	Присвоенное имя
42–46	Блеск (зв. вел.)
106–113	Эпоха, для которой определены элементы орбиты (yyyymmdd)
115–124	Средняя аномалия [°]
126–135	Аргумент долготы (J2000) [°]
137–146	Долгота восходящего узла (J2000) [°]
148–156	Наклонение орбиты (J2000) [°]
158–167	Эксцентриситет
169–180	Большая полуось [а. е.]

Чтобы упростить использование этой базы данных, можно пользоваться программой AOEcat, которая извлекает из файла данные об астероиде по указанному номеру или имени.

```
  AOEcat: Selection of Data from the Asteroid Orbital Elements Database
               (c) 1999 Oliver Montenbruck, Thomas Pfleger

 Name/number of the minor planet: Lutetia

 2000 02 26.0   ! Epoch       Minor planet(21) Lutetia
   2.43561876   ! a [AU]
   0.16168474   ! e
     3.066091   ! i [deg]
    80.947035   ! Omega [deg]
   250.127264   ! omega [deg]
   328.800319   ! M [deg]
       2000.0   ! Equinox
```

Выходной формат выбран таким, чтобы полученные данные можно было без обработки передавать программе Numint. С этой же целью при вызове программ AOEcat в командной строке можно указать имя выходного файла (см. раздел А.1.3). При вводе номера малой планеты следите за тем, чтобы до и после номера не было пробелов. Напротив, при задании имен (например, Tycho Brahe) и других обозначений (например, 5073 T-3) пробелы на соответствующих позициях необходимы. Буквы можно вводить как на верхнем, так на нижнем регистре. Полный исходный текст программы AOEcat вместе со всеми остальными программами можно найти на прилагаемом CD-ROM.

6. Планетные орбиты

За исключением Меркурия и Плутона орбиты больших планет весьма сходны между собой. Все они имеют малые эксцентриситеты и небольшие наклонения к плоскости эклиптики. Это отличает их от кометных орбит, которые расположены в пространстве случайным образом и чья форма меняется в широком диапазоне. Размеры почти круговых планетных орбит довольно значительно различаются, и поэтому они никогда не испытывают тесных сближений.

Из этих свойств планетных орбит вытекают два следствия, важных для расчета эфемерид. Первое касается использования невозмущенных кеплеровых орбит (см. главу 4) для общего описания формы орбиты. Благодаря малому эксцентриситету истинная аномалия испытывает лишь незначительные периодические отклонения от средней аномалии. Поэтому, пользуясь разложениями в ряды, ее можно вычислить быстрее, чем решая уравнение Кеплера. То же самое относится и к расстоянию от Солнца, которое обычно меняется в пределах примерно одного процента величины большой полуоси орбиты. Наконец, для вычисления расстояния планеты от плоскости эклиптики также можно воспользоваться быстро сходящимся итерационным алгоритмом, избегая сложной процедуры вычисления векторов Гаусса. В этой главе мы подробно рассмотрим все эти три аппроксимации.

Второе следствие касается изменения планетных орбит в результате их взаимного гравитационного притяжения. Поскольку планеты не испытывают тесных сближений, это притяжение всегда на несколько порядков меньше солнечной гравитации. Орбиты планет никогда не испытывают таких значительных изменений, как это бывает с кометами после сближения с Юпитером или Сатурном. В результате их можно описать средними кеплеровыми эллипсами, на которые наложены небольшие периодические возмущения.

На самом деле, эфемериды планет вычисляются в наше время с использованием методов численного интегрирования, а надлежащим образом отформатированные результаты сохраняют на магнитных носителях. Современные компьютеры благодаря своей мощности сделали этот простой метод поразительно точным. Однако у него есть и некоторые недостатки. В частности, у нас теперь нет аналитического описания движения планет, как в случае решения «чистой» задачи двух тел. Одним из величайших достижений небесной механики было и остается построение рядов, позволяющих на любой момент времени определить отклонение планеты от невозмущенной орбиты. Получение этих рядов, как легко понять, взглянув на количество членов в них, выходит далеко за рамки нашей книги. Однако для наших целей, то есть для точного вычисления положений планет, достаточно лишь в общих чертах понимать структуру этих рядов и владеть способами

их численного суммирования. Тех же, кто хочет более подробно разобраться, как они были получены, мы адресуем к различным учебникам небесной механики.

Одно из самых известных аналитических описаний движения планет — это таблицы движения внутренних планет от Меркурия до Марса, составленные Симоном Ньюкомбом в Американской морской обсерватории (U.S. Naval Observatory). Хотя они были созданы еще в конце XIX века, до 1984 года они оставались основой для составления эфемерид «Астрономического альманаха». Аналогичные таблицы для Юпитера и Сатурна были составлены Г. У. Хиллом, а для Урана и Нептуна — Ньюкомбом, однако вскоре вместо них стали использовать эфемериды, полученные численным интегрированием, поскольку эти таблицы не обеспечивали на длительных отрезках времени такой же точности, как таблицы для внутренних планет.

В основу данной главы положены перечисленные работы, а также ряды для Плутона, построенные Е. Гоффином, Дж. Меёсом и К. Стюартом. Формулы пертурбационных членов для разных планет иногда настолько сильно различаются, что их невозможно по единой схеме непосредственно преобразовать в программу. Поэтому все ряды были предварительно обработаны программой аналитических вычислений. Это позволило разделить кеплерово движение и пертурбационные члены, которые в исходных рядах были смешаны и описывались суперпозицией периодических движений в долготе, широте и радиусе. Кроме того, было исключено значительное число малых членов (в основном <1″), которые практически не оказывают влияния на точность вычисления наблюдаемых положений. Благодаря всем этим преобразованиям удалось получить довольно компактный код без существенных потерь точности вычислений.

6.1. Разложение в ряд уравнения Кеплера

Невозмущенное движение планеты по кеплерову эллипсу описывается строго периодической функцией средней аномалии M. Каждый раз, когда средняя аномалия меняется на 360°, планета совершает один полный оборот и возвращается в исходную точку. Таким образом, любая функция положения на орбите может быть представлена рядом Фурье, то есть суммой синусов и косинусов величин, кратных M. Примером такого представления может служить ряд

$$\nu - M = a_1 \sin(M) + a_2 \sin(2M) + a_3 \sin(3M) + \ldots,$$

выражающий разность между истинной и средней аномалиями (называемую также как *уравнением центра). Другой пример* — ряд для расстояния r планеты от притягивающего центра (в единицах большой полуоси a):

$$r/a = b_0 + b_1 \cos(M) + b_2 \cos(2M) + \ldots .$$

Коэффициенты a_i и b_i в этих рядах являются функциями эксцентриситета орбиты e. Наиболее важные члены ряда для $\nu - M$ и r/a (до второго порядка по эксцентриситету) выглядят так:

$$\nu - M = 2e\sin(M) + \frac{5}{4}e^2 \sin(2M) + \ldots \text{ (радианы)} \tag{6.1}$$

и

$$r/a = \left(1 + \frac{1}{2}e^2\right) - e\cos(M) - \frac{1}{2}e^2\cos(2M) + \ldots . \qquad (6.2)$$

Для малых значений эксцентриситета оба этих ряда сходятся очень быстро, поскольку каждый следующий член содержит более высокую степень малой величины *e*. Для земной орбиты $e = 0{,}0167$, и мы получаем

$$\nu = M + 0{,}0334\sin(M) + 0{,}00035\sin(2M) + \ldots \text{ (радианы)} =$$
$$= M + 1{,}^\circ916\sin(M) + 0{,}^\circ020\sin(2M) + \ldots,$$
$$r = a\cdot(1{,}00014 - 0{,}0167\cos(M) - 0{,}00014\cos(2M) + \ldots).$$

Погрешность этих сокращенных формул составляет всего около одной угловой секунды или примерно 10^{-5} а. е. и легко может быть уменьшена, есть включить дополнительные члены.

Поскольку полный вывод этих формул требует довольно глубокого владения используемым математическим аппаратом, мы ограничимся тем, что продемонстрируем вывод только первого члена уравнения центра. Начнем с уравнения Кеплера (4.7):

$$E - e\sin E = M.$$

Поскольку значения E и M различаются на величину порядка e, вычисляя разность между эксцентирической и истинной аномалиями, мы можем в качестве хорошего приближения использовать $e\sin M$ вместо $e\sin E$:

$$E - M \approx e\sin M.$$

Кроме того, из уравнения конического сечения $r = a(1-e^2)/(1+e\cos\nu)$, где $r\cos\nu = a(\cos E - e)$, получаем

$$\cos\nu \approx \cos E - e(1 - \cos\nu\cos E).$$

Во втором члене правой части уравнения мы вновь можем подставить $\cos E$ вместо $\cos\nu$:

$$\cos\nu \approx \cos E - \{e\sin E\}\sin E.$$

Если сравнить это выражение с рядом Тейлора для косинуса

$$\cos(x + \Delta x) \approx \cos x - \Delta x\sin x,$$

то нетрудно заметить, что при малых эксцентриситетах разность $\nu - E$ примерно равна $e\sin E$. Таким образом, первый член в уравнении центра будет равен

$$\nu - M = (\nu - E) + (E - M) \approx e\sin E + e\sin M \approx 2e\sin M.$$

Для малых значений наклонения орбиты i движение всегда происходит вблизи эклиптики. Поэтому эклиптическая долгота планеты l отличается от суммы $\varpi + \nu$ долготы перигелия и истинной аномалии только на величину небольшой

поправки R, которую называют *редукцией к эклиптике*. Поправка R исчезает при $i=0$, а для малых наклонений в первом приближении имеет значение

$$R = l-(\nu+\varpi) \approx -\operatorname{tg}^2\left(\frac{i}{2}\right)\cdot\sin(2u) \text{ (радианы)}, \tag{6.3}$$

где $u = \nu+\omega = \nu+\varpi-\Omega$. Учитывая, что $\nu \approx M$, R можно выразить как

$$R = -\operatorname{tg}^2\left(\frac{i}{2}\right)\cdot\{\sin(2\omega)\cos(2M)+\cos(2\omega)\sin 2M)\}.$$

Значение R в градусах можно получить, умножив эту величину на $180°/\pi$.

С эклиптической широтой b планеты можно поступить аналогичным образом. Взяв уравнение (4.20)

$$z = r\cdot(\sin(\nu+\omega)\sin(i)), \tag{6.4}$$

представляющее z-координату как функцию элементов орбиты, и учитывая, что для малых значений наклонения i и эклиптической широты b

$$\sin(b) = z/r, \tag{6.5}$$

получаем приближение

$$b \approx i\sin(\nu+\omega) = \{i\sin\omega\}\cos(\nu)+\{i\cos\omega\}\sin(\nu).$$

Если мы заменим ν на $\nu \approx M + 2e\sin M$, то разложение в ряд Тейлора даст нам

$$\cos\nu \approx \cos M-(2e\sin M)\sin M = \cos(M)-e(1-\cos(2M)),$$
$$\sin\nu \approx \sin M+(2e\sin M)\cos M = \sin(M)+e\sin(2M),$$

откуда получаем

$$b = \{i\sin\omega\}\cdot\{\cos(M)-e(1-\cos(2M))\} +$$
$$+\{i\cos\omega\}\cdot\{\sin(M)+e\sin(2M)\}.$$

Таким образом, наиболее важные члены описания невозмущенной планетной орбиты имеют следующий вид:

$$\begin{aligned} l = {} & \varpi + M \\ & +180°/\pi\cdot\{2e\}\cdot\sin(M) \\ & +180°/\pi\cdot\left\{\frac{5}{4}e^2-\operatorname{tg}^2\left(\frac{i}{2}\right)\cos(2\omega)\right\}\cdot\sin(2M) \\ & +180°/\pi\cdot\left\{-\operatorname{tg}^2\left(\frac{i}{2}\right)\sin(2\omega)\right\}\cdot\cos(2M) \end{aligned} \tag{6.6}$$

$$\begin{aligned} b = {} & -\{ie\sin\omega\} \\ & +\{i\sin\omega\}\cos(M) \quad +\{i\cos\omega\}\sin(M) \\ & +\{ie\sin\omega\}\cos(2M)+\{ie\cos\omega\}\sin(2M) \end{aligned} \tag{6.7}$$

$$r = \left\{a(1 + e^2/2)\right\} - \left\{ae\right\} \cdot \cos(M) - \left\{ae^2/2\right\} \cdot \cos(2M). \qquad (6.8)$$

Подставляя в эти уравнения значения элементов орбиты, мы получаем очень компактные формулы для вычисления эклиптических координат (l, b, r), как функций средней аномалии M. Взяв, например, данные для Юпитера

$$\begin{array}{lll} a = 5{,}2\ \text{а. е.} & \Omega = 100\overset{\circ}{,}0 & \omega = 274\overset{\circ}{,}0 \\ e = 0{,}048 & i = 1\overset{\circ}{,}31 & \varpi = 14\overset{\circ}{,}0, \end{array}$$

получаем следующие ряды:

$$\begin{aligned} l &= M + 14\overset{\circ}{,}0 + 19800'' \sin(M) + 620'' \sin(2M) + 4'' \cos(2M), \\ b &= +226'' - 4700'' \cos(M) + 329'' \sin(M) - 226'' \cos(2M) + 16'' \sin(2M), \\ r &= (5{,}206 - 0{,}250 \cos(M) - 0{,}006 \cos(2M))\ \text{а. е.} \end{aligned}$$

6.2. Возмущающие члены

Взаимные возмущения движения планет, которые приводят к отклонениям от кеплеровых орбит, представляются дополнительными членами в рядах, содержащими аномалии обеих планет — возмущаемой и возмущающей. В дальнейшем мы будем обозначать M_n среднюю аномалию n-й планеты. Например, M_5 относится к Юпитеру, M_6 — к Сатурну:

$$\begin{array}{llr} M_1 = 0\overset{r}{,}4855407 + & 415\overset{r}{,}2014314 \cdot T & \text{(Меркурий)} \\ M_2 = 0\overset{r}{,}1400197 + & 162\overset{r}{,}5494552 \cdot T & \text{(Венера)} \\ M_3 = 0\overset{r}{,}9931266 + & 99\overset{r}{,}9973604 \cdot T & \text{(Солнце, Земля)} \\ M_4 = 0\overset{r}{,}0538553 + & 53\overset{r}{,}1662736 \cdot T & \text{(Марс)} \\ M_5 = 0\overset{r}{,}0565314 + & 8\overset{r}{,}4302963 \cdot T & \text{(Юпитер)} \\ M_6 = 0\overset{r}{,}8829867 + & 3\overset{r}{,}3947688 \cdot T & \text{(Сатурн)} \\ M_7 = 0\overset{r}{,}3967117 + & 1\overset{r}{,}1902849 \cdot T & \text{(Уран)} \\ M_8 = 0\overset{r}{,}7214906 + & 0\overset{r}{,}6068526 \cdot T & \text{(Нептун)} \\ M_9 = 0\overset{r}{,}0385795 + & 0\overset{r}{,}4026667 \cdot T & \text{(Плутон)} \end{array} \qquad (6.9)$$

Приведенные значения выражены в полных оборотах (1^r)[1]. Если домножить их соответственно на 360° или на 2π, то мы получим значения средних аномалий соответственно в градусах или радианах. Время, как обычно, измеряется в юлианских столетиях от эпохи J2000:

$$T = (\text{JD} - 2451545)/35625.$$

[1] В некоторых случаях в программах использованы значения, немного отличающиеся от приведенных здесь. Это не ошибка, а следствие различия в теориях, лежащих в основе полученных разложений.

Таблица 6.1. Периодические члены орбиты Юпитера

			$dl['']$		$dr[10^5$ а. е.$]$		$db['']$		
i_5		i_T	cos	sin	cos	sin	cos	sin	
1		0	−113,1	19998,6	−25208,2	−142,2	−4670,7	288,9	Кеплеровские
1		1	−76,1	66,9	−84,2	−95,8	21,6	29,4	члены
1		2	−0,5	−0,3	0,4	−0,7	0,1	−0,1	
2		0	−3,4	632,0	−610,6	−6,5	−226,8	12,7	
2		1	−4,2	3,8	−4,1	−4,5	0,2	0,6	
3		0	−0,1	28,0	−22,1	−0,2	−12,5	0,7	
4		0	0,0	1,4	−1,0	0,0	−0,6	0,0	
i_5	i_6	i_T							
−1	−1	0	−0,2	1,4	2,0	0,6	0,1	−0,2	Сатурн
0	−1	0	9,4	8,9	3,9	−8,3	−0,4	−1,4	
0	−2	0	5,6	−3,0	−5,4	−5,7	−2,0	0,0	
0	−3	0	−4,0	−0,1	0,0	5,5	0,0	0,0	
0	−5	0	3,3	−1,6	−1,6	−3,1	−0,5	−1,2	
1	−1	0	78,8	−14,5	11,5	64,4	−0,2	0,2	
1	−2	0	−2,0	−132,4	28,8	4,3	−1,7	0,4	
1	−2	1	−1,1	−0,7	0,2	−0,3	0,0	0,0	
1	−3	0	−7,5	−6,8	−0,4	−1,1	0,6	−0,9	
1	−4	0	0,7	0,7	0,6	−1,1	0,0	−0,2	
1	−5	0	51,5	−26,0	−32,5	−64,4	−4,9	−12,4	
1	−5	1	−1,2	−2,2	−2,7	1,5	−0,4	0,3	
2	−1	0	5,3	−0,7	0,7	6,1	0,2	1,1	
2	−2	0	−76,4	−185,1	260,2	−108,0	1,6	0,0	
2	−3	0	66,7	47,8	−51,4	69,8	0,9	0,3	
2	−3	1	0,6	−1,0	1,0	0,6	0,0	0,0	
2	−4	0	17,0	1,4	−1,8	9,6	0,0	−0,1	
2	−5	0	1066,2	−518,3	−1,3	−23,9	1,8	−0,3	
2	−5	1	−25,4	−40,3	−0,9	0,3	0,0	0,0	
2	−5	2	−0,7	0,5	0,0	0,0	0,0	0,0	
3	−2	0	−5,0	−11,5	11,7	−5,4	2,1	−1,0	
3	−3	0	16,9	−6,4	13,4	26,9	−0,5	0,8	
3	−4	0	7,2	−13,3	20,9	10,5	0,1	−0,1	
3	−5	0	68,5	134,3	−166,9	86,5	7,1	15,2	*
3	−5	1	3,5	−2,7	3,4	4,3	0,5	−0,4	
3	−6	0	0,6	1,0	−0,9	0,5	0,0	0,0	
3	−7	0	−1,1	1,7	−0,4	−0,2	0,0	0,0	
4	−2	0	−0,3	−0,7	0,4	−0,2	0,2	−0,1	
4	−3	0	1,1	−0,6	0,9	1,2	0,1	0,2	
4	−4	0	3,2	1,7	−4,1	5,8	0,2	0,1	
4	−5	0	6,7	8,7	−9,3	8,7	−1,1	1,6	
4	−6	0	1,5	−0,3	0,6	2,4	0,0	0,0	
4	−7	0	−1,9	2,3	−3,2	−2,7	0,0	−0,1	
4	−8	0	0,4	−1,8	1,9	0,5	0,0	0,0	
4	−9	0	−0,2	−0,5	0,3	−0,1	0,0	0,0	
4	−10	0	−8,6	−6,8	−0,4	0,1	0,0	0,0	
4	−10	1	−0,5	0,6	0,0	0,0	0,0	0,0	
5	−5	0	−0,1	1,5	−2,5	−0,8	−0,1	0,1	
5	−6	0	0,1	0,8	−1,6	0,1	0,0	0,0	
5	−9	0	−0,5	−0,1	0,1	−0,8	0,0	0,0	
5	−10	0	2,5	−2,2	2,8	3,1	0,1	−0,2	
i_5	i_7	i_T							
1	−1	0	0,4	0,9	0,0	0,0	0,0	0,0	Уран
1	−2	0	0,4	0,4	−0,4	0,3	0,0	0,0	
i_5	i_6	i_7							
2	−6	3	−0,8	8,5	−0,1	0,0	0,0	0,0	Сатурн
3	−6	3	0,4	0,5	−0,7	0,5	−0,1	0,0	и Уран

Удобнее всего представлять совокупность членов ряда, описывающего возмущения, в виде таблицы. Например, коэффициенты разложений для Юпитера представленыв табл. 6.1. В нее включены члены, соответствующие периодическому движению в эклиптической долготе (dl) и широте (db), а также в расстоянии от Солнца (dr).Члены тригонометрических рядов с одинаковыми аргументами для удобства дальнейших вычислений сведены в одну строку. В развернутой форме выражения, соответствующие, например, строке, отмеченной звездочкой (*), имеют вид:

$$dl = 68''\!,5 \cdot \cos(3M_5 - 5M_6) + 134''\!,3 \cdot \sin(3M_5 - 5M_6),$$

$$dr = (-166{,}9 \cdot \cos(3M_5 - 5M_6) + 86{,}5 \cdot \sin(3M_5 - 5M_6)) \cdot 10^{-5}\ \text{а. е.},$$

$$db = 7''\!,1 \cdot \cos(3M_5 - 5M_6) + 15''\!,2 \cdot \sin(3M_5 - 5M_6).$$

Эти значения следует добавить в качестве поправок к долготе, широте и радиусу. Члены, которые должны дополнительно умножаться на T или T^2, отмечены цифрами 1 и 2 в колонке i_T.

Влияние планетных возмущений не укладывается полностью в периодическую схему, но вызывает также *вековые* изменения средних значений орбитальных элементов. Это означает, что зависимость от времени T появляется даже в членах, описывающих чисто кеплеровское движение. В функции `JupPosition` для вычисления координат Юпитера используются следующие уравнения:

$$l = M_5 + 14^\circ\!,00076 + (5025''\!,2 + 0''\!,8T)T + dl,$$

$$b = 227''\!,3 - 0''\!,3T + db,$$

$$r = (5{,}208873 + 0{,}000041T)\ \text{а. е.} + dr,$$

где dl, db и dr — полные периодические составляющие орбитального движения, включающие кеплеровы члены и возмущения со стороны Сатурна и Урана. Значения эклиптической долготы и широты, определенные по этим формулам, отнесены к *среднему равноденствию даты*. Формула для l содержит большой вековой член $5025''\!,2 \cdot T$, отражающий движение точки весеннего равноденствия.

Размеры книги не позволяют нам привести коэффициенты рядов, учитывающих возмущения для всех планет. Но поскольку они имеют однотипную структуру приведенных данных для Юпитера должно быть достаточно, чтобы разобраться в формулах, использованных при составлении программ.

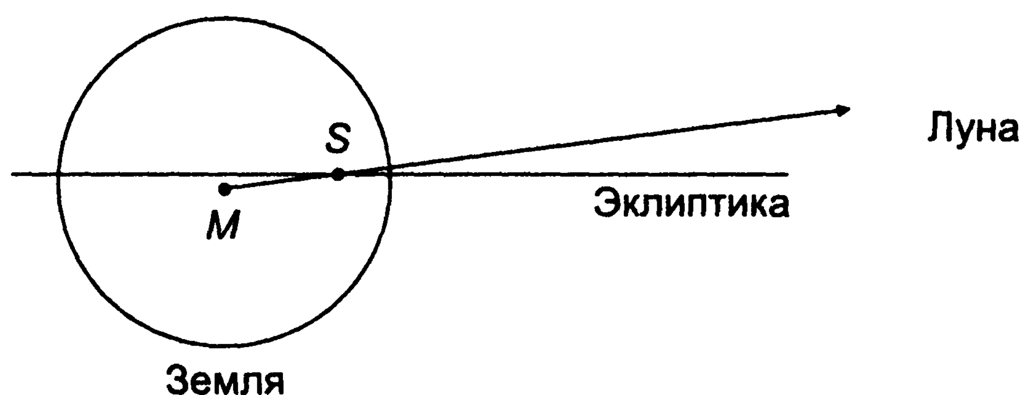


Рис. 6.1. Положение центра масс Земли и Луны

Некоторые отличия есть только в способе вычисления движения Земли. Земля обращается вокруг Солнца вместе с Луной, и в первом приближении можно

считать, что оба небесных тела расположены в своем центре масс (барицентре). Движение барицентра (которое, кстати, определяет положение эклиптики) можно описать возмущенной кеплеровой орбитой, подобно тому, как это делалось для других планет. Для описания движения по этой орбите можно построить соответствующие ряды. С учетом соотношения масс Земли и Луны, расстояние Земли от барицентра в 81 раз меньше расстояния от Земли до Луны и составляет, таким образом, примерно 3/4 радиуса Земли (рис. 6.1). Если наблюдать с Солнца, эклиптическая долгота Земли испытывает колебания относительно среднего значения на величину до 7 угловых секунд. В то же время эклиптическая широта Земли может варьироваться в пределах половины угловой секунды.

6.3. Численное суммирование рядов

Для того чтобы достичь точности порядка нескольких угловых секунд, необходимо учитывать сотни возмущающих членов для каждой планеты. Если последовательно запрограммировать вычисление всех этих членов, то мы получим не только очень длинную и некрасивую, но еще и очень медленно работающую программу. Дело в том, что вычисление тригонометрических функций является довольно сложной задачей и по сравнению с такими операциями, как сложение и умножение, выполняется относительно медленно. Поскольку ряды содержат большое количество тригонометрических функций, затраты времени становятся заметными даже при использовании мощных компьютеров.

Значительной части этой работы можно избежать, воспользовавшись теоремами о синусе суммы и косинусе суммы:

$$\begin{aligned} \cos(\alpha_1 + \alpha_2) &= \cos\alpha_1 \cos\alpha_2 - \sin\alpha_1 \sin\alpha_2, \\ \sin(\alpha_1 + \alpha_2) &= \sin\alpha_1 \cos\alpha_2 + \cos\alpha_1 \sin\alpha_2. \end{aligned} \tag{6.10}$$

Эти формулы позволяют вычислять тригонометрические функции суммы и разности углов по значениям этих функций для отдельных углов, составляющих сумму или разность. Для заданной величины M можно определить значения $\cos M$ и $\sin M$, а затем, пользуясь формулами (6.10), вычислить и запомнить $\cos(iM)$ и $\sin(iM)$ для различных значений $i = 2, 3, \ldots$ Начнем с того, что реализуем функцию AddThe:

```
//------------------------------------------------------------------------
// AddThe: Вычисляет cos(alpha+beta) и sin(alpha+beta),
//         используя теоремы о косинусе суммы и синусе суммы
//   c1, s1    cos(alpha), sin(alpha)
//   c2, s2    cos(beta), sin(beta)
//   c, s      cos(alpha+beta), sin(alpha+beta)
//------------------------------------------------------------------------
void AddThe ( double c1, double s1, double c2, double s2,
              double& c, double& s )
{
  c = c1 * c2 - s1 * s2;
  s = s1 * c2 + c1 * s2;
}
```

Все дальнейшие операции, а также необходимые вспомогательные массивы для хранения предварительно вычисленных значений собраны в отдельном классе Pert. Две наиболее важные его функции называются Init и Term. Если мы обозначим M среднюю аномалию возмущаемой планеты, а m среднюю аномалию возмущающей планеты, тогда функция Init вычисляет $\sin(IM+\phi)$ и $\cos(IM+\phi)$ для $I_{\min} \le I \le I_{\max}$, а также $\sin(im)$ и $\cos(im)$ для $i_{\min} \le i \le i_{\max}$. Здесь ϕ — вспомогательное значение, которое позволяет единообразно обрабатывать члены, зависящие от средней аномалии третьей планеты. При вычислении отдельных возмущающих членов метод Term использует ранее определенные значения тригонометрических функций и суммирует вклады в долготу, радиус и широту в соответствующих членах класса. Накопленные значения можно затем получить при помощи методов dl, dr и db.

```
// Константы
const int o   = 16;         // Диапазон значений индекса
const int dim = 2*o+1;      // Размер рабочего массива

// Определение класса Pert для суммирования тригонометрических рядов
// планетных возмущений
class Pert
{
  public:
    // Задает время, среднюю аномалию и диапазон значений индекса
    void Init ( double T,
                double M, int I_min, int I_max,
                double m, int i_min, int i_max,
                double phi=0.0 );
    // Суммирует возмущения в долготе, радиусе и широте
    void Term ( int I, int i, int iT,
                double dlc, double dls,
                double drc, double drs,
                double dbc, double dbs );
    // Возвращают возмущения в долготе, радиусе и широте
    double dl();
    double dr();
    double db();
  private:
    double m_T;
    double m_cosM, m_sinM;
    double m_C[dim], m_S[dim], m_c[dim], m_s[dim],
    double m_dl, m_db, m_dr;
    double m_u, m_v;
};

// Задает время, среднюю аномалию и диапазон значений индекса
void Pert::Init ( double T,
                  double M, int I_min, int I_max,
                  double m, int i_min, int i_max,
                  double phi )
{
  int i;
  m_dl=0.0; m_dr=0.0; m_db=0.0; // сброс ранее накопленных значений
  m_T=T;                         // установка времени
  // cos и sin для значений, кратных M
```

```
  m_cosM=cos(M); m_sinM=sin(M);
  m_C[o]=cos(phi);
  m_S[o]=sin(phi);
  for (i=0; i<I_max; i++)
    AddThe ( m_C[o+i],m_S[o+i], +m_cosM,+m_sinM, m_C[o+i+1],m_S[o+i+1] );
  for (i=0; i>I_min; i--)
    AddThe ( m_C[o+i],m_S[o+i], +m_cosM,-m_sinM, m_C[o+i-1],m_S[o+i-1] );
  // cos и sin для значений, кратных m
  m_c[o]=1.0; m_c[o+1]=cos(m); m_c[o-1]=+m_c[o+1];
  m_s[o]=0.0; m_s[o+1]=sin(m); m_s[o-1]=-m_s[o+1];
  for (i=1; i<i_max; i++)
    AddThe ( m_c[o+i],m_s[o+i], m_c[o+1],m_s[o+1], m_c[o+i+1],m_s[o+i+1] );
  for (i=-1; i>i_min; i--)
    AddThe ( m_c[o+i],m_s[o+i], m_c[o-1],m_s[o-1], m_c[o+i-1],m_s[o+i-1] );
}

// Суммирует возмущения в долготе, радиусе и широте
void Pert::Term ( int I, int i, int iT,
                  double dlc, double dls,
                  double drc, double drs,
                  double dbc, double dbs )
{
  if (iT == 0)
    AddThe ( m_C[o+I],m_S[o+I], m_c[o+i],m_s[o+i], m_u,m_v );
  else
    { m_u *= m_T; m_v *= m_T; };
  m_dl += ( dlc*m_u + dls*m_v );
  m_dr += ( drc*m_u + drs*m_v );
  m_db += ( dbc*m_u + dbs*m_v );
}

// Возвращают возмущения в долготе, радиусе и широте
double Pert::dl() { return m_dl; }
double Pert::dr() { return m_dr; }
double Pert::db() { return m_db; }
```

Приводимый ниже пример функции JupiterPos демонстрирует, как вычисляются и суммируются все возмущающие члены. Метод Term вычисляет значения $u = \cos(i_5 M_5 + i_j M_j)$ и $v = \sin(i_5 M_5 + i_j M_j)$ для индексов i_5 и i_j, домножает их на соответствующие коэффициенты и добавляет к значениям *dl*, *dr* и *db*. Если допустить, что члены с равными значениями i_5 и i_j, но разными $i_T = 0, 1, \ldots$ вычисляются последовательно, то достаточно (и именно так сделано в функции Term) вычислять u и v только для $i_T = 0$, а затем просто домножать значение на T для получения $i_T = 1$ или $i_T = 2$. Хотя на первый взгляд кажется, что такой прием увеличивает объем работы, однако при большом числе членов, как в данном случае, это скоро окупается.

```
//------------------------------------------------------------------------
// JupiterPos: Вычисляет положение Юпитера, суммируя ряды
//   T         Время в юлианских столетиях, считая от J2000
// <return>:   Гелиоцентрическое положение [а.е.]
//             относительно эклиптики и равноденствия даты
//------------------------------------------------------------------------
Vec3D JupiterPos (double T)
{
```

```
// Переменные
double  M5,M6,M7;                // Средние аномалии
Pert    Sat,Ura;                 // Возмущения
double  phi,c,s;
double  dl, dr, db;              // Поправки в долготе ["], радиусе [a.e.] и широте ["]
double  l,b,r;                   // Эклиптические координаты

// Средние аномалии планет в [рад]
M5 = pi2 * Frac ( 0.0565314 + 8.4302963*T );
M6 = pi2 * Frac ( 0.8829867 + 3.3947688*T );
M7 = pi2 * Frac ( 0.3969537 + 1.1902586*T );

// Кеплерово движение и возмущения Сатурна
Sat.Init ( T, M5,-1,5, M6,-10,0 );
Sat.Term (-1, -1,0,  -0.2,    1.4,     2.0,   0.6,    0.1, -0.2);
Sat.Term ( 0, -1,0,   9.4,    8.9,     3.9,  -8.3,   -0.4, -1 4);
Sat Term ( 0, -2,0,   5.6,   -3.0,    -5.4,  -5.7,   -2.0,  0.0);
Sat.Term ( 0, -3,0,  -4.0,   -0.1,     0.0,   5.5,    0.0,  0.0);
Sat.Term ( 0, -5,0,   3.3,   -1.6,    -1.6,  -3.1,   -0.5, -1.2);
Sat.Term ( 1,  0,0,-113.1,19998.6,-25208.2,-142.2,-4670.7,288.9);
Sat.Term ( 1,  0,1, -76.1,   66.9,   -84.2, -95.8,   21.6, 29.4);
Sat.Term ( 1,  0,2,  -0 5,   -0.3,     0.4,  -0.7,    0.1, -0 1),
Sat.Term ( 1, -1,0,  78.8,  -14.5,    11.5,  64.4,   -0.2,  0.2);
Sat.Term ( 1, -2,0,  -2.0, -132.4,    28.8,   4.3,   -1.7,  0.4),
Sat.Term ( 1, -2,1,  -1.1,   -0.7,     0.2,  -0.3,    0.0,  0.0);
Sat.Term ( 1, -3,0,  -7.5,   -6.8,    -0.4,  -1.1,    0.6, -0.9);
Sat.Term ( 1, -4,0,   0.7,    0.7,     0.6,  -1.1,    0.0, -0.2);
Sat.Term ( 1, -5,0,  51.5,  -26.0,   -32.5, -64.4,   -4.9,-12.4),
Sat.Term ( 1, -5,1,  -1.2,   -2.2,    -2.7,   1.5,   -0.4,  0.3);
Sat.Term ( 2,  0,0,  -3.4,  632.0,  -610.6,  -6.5, -226.8, 12.7);
Sat.Term ( 2,  0,1,  -4.2,    3.8,    -4.1,  -4.5,    0.2,  0.6);
Sat.Term ( 2, -1,0,   5 3,   -0.7,     0.7,   6.1,    0.2,  1.1);
Sat.Term ( 2, -2,0, -76.4, -185.1,   260.2,-108.0,    1.6,  0.0);
Sat.Term ( 2, -3,0,  66.7,   47.8,   -51 4,  69.8,    0.9,  0.3),
Sat.Term ( 2, -3,1,   0.6,   -1.0,     1.0,   0 6,    0 0,  0 0);
Sat.Term ( 2, -4,0,  17.0,    1.4,    -1.8,   9.6,    0.0, -0 1),
Sat.Term ( 2, -5,0,1066.2, -518.3,    -1.3, -23.9,    1.8, -0.3);
Sat.Term ( 2, -5,1, -25.4,  -40 3,    -0.9,   0.3,    0.0,  0.0);
Sat.Term ( 2, -5,2,  -0.7,    0.5,     0.0,   0 0,    0.0,  0.0);
Sat.Term ( 3,  0,0,  -0.1,   28 0,   -22.1,  -0.2,  -12.5,  0.7),
Sat.Term ( 3, -2,0,  -5.0,  -11.5,    11.7,  -5.4,    2.1, -1.0);
Sat.Term ( 3, -3,0,  16.9,   -6.4,    13.4,  26.9,   -0.5,  0.8);
Sat.Term ( 3, -4,0,   7.2,  -13 3,    20.9,  10.5,    0.1, -0.1);
Sat.Term ( 3, -5,0,  68.5,  134.3,  -166.9,  86.5,    7 1, 15.2);
Sat.Term ( 3, -5,1,   3.5,   -2 7,     3.4,   4.3,    0.5, -0.4);
Sat.Term ( 3, -6,0,   0.6,    1.0,    -0.9,   0 5,    0.0,  0.0);
Sat.Term ( 3, -7,0,  -1.1,    1.7,    -0.4,  -0 2,    0.0,  0.0),
Sat.Term ( 4,  0,0,   0.0,    1.4,    -1.0,   0.0,   -0.6,  0.0);
Sat.Term ( 4, -2,0,  -0.3,   -0.7,     0.4,  -0.2,    0.2, -0.1);
Sat.Term ( 4, -3,0,   1.1,   -0.6,     0.9,   1.2,    0.1,  0 2);
Sat.Term ( 4, -4,0,   3.2,    1.7,    -4.1,   5.8,    0.2,  0.1),
Sat.Term ( 4, -5,0,   6.7,    8.7,    -9.3,   8.7,   -1.1,  1.6),
Sat.Term ( 4, -6,0,   1.5,   -0.3,     0.6,   2.4,    0.0,  0.0),
Sat.Term ( 4, -7,0,  -1.9,    2.3,    -3.2,  -2.7,    0.0, -0.1);
Sat.Term ( 4, -8,0,   0.4,   -1.8,     1.9,   0.5,    0.0,  0.0);
Sat Term ( 4, -9,0,  -0.2,   -0.5,     0.3,  -0.1,    0 0,  0.0);
Sat.Term ( 4,-10,0,  -8 6,   -6.8,    -0.4,   0.1,    0.0,  0.0);
```

```
  Sat Term ( 4,-10,1,  -0.5,    0.6,     0.0,   0.0,    0.0,  0.0);
  Sat.Term ( 5, -5,0,  -0.1,    1.5,    -2.5,  -0.8,   -0 1,  0 1);
  Sat.Term ( 5, -6,0,   0.1,    0.8,    -1.6,   0.1,    0 0,  0.0);
  Sat.Term ( 5, -9,0,  -0.5,   -0.1,     0.1,  -0 8,    0 0,  0 0);
  Sat.Term ( 5,-10,0,   2.5,   -2.2,     2.8,   3.1,    0.1, -0.2);
  dl = Sat.dl();  dr = Sat.dr();  db = Sat.db();

  // Возмущения Урана
  Ura.Init ( T, M5,1,1, M7,-2,-1 ),
  Ura.Term ( 1, -1,0,   0 4,    0.9,     0.0,   0.0,    0.0,  0 0);
  Ura.Term ( 1, -2,0,   0.4,    0.4,    -0.4,   0.3,    0.0,  0 0);
  dl += Ura.dl();  dr += Ura.dr();  db += Ura.db();

  // Возмущения Сатурна и Урана
  phi = (2*M5-6*M6+3*M7); c=cos(phi); s=sin(phi);
  dl += -0.8*c+8.5*s;  dr += -0.1*c;
  phi = (3*M5-6*M6+3*M7); c=cos(phi); s=sin(phi),
  dl += +0.4*c+0.5*s;  dr += -0.7*c+0.5*s;  db += -0 1*c;

  // Эклиптические координаты ([рад],[а.е.])
  l = pi2 * Frac (0.0388910 + M5/pi2 + ( (5025.2+0 8*T)*T + dl ) / 1296.0e3);
  r = 5.208873 + 0.000041*T  +  dr * 1.0E-5;
  b = ( 227.3 - 0.3*T + db ) / Arcs;
  return Vec3D ( Polar(l,b,r) );  // Вектор положения
}
```

Для заданного момента времени T функция `JupiterPos` и аналогичные ей функции `MercuryPos`, `VenusPos`, ..., `PlutoPos` возвращают значения гелиоцентрических долготы l, широты b, а также расстояние r до соответствующей планеты, приведенные к равноденствию указанной даты. Время измеряется в юлианских столетиях от эпохи J2000:

$$T = (\mathrm{JD} - 2451545)/36525.$$

Обратите внимание, что при вычислении юлианской даты JD следует использовать эфемеридное время ET (или динамическое время TDT/TDB), но не Всемирное время UT (см. раздел 3.4). Процедуры для вычисления геоцентрических солнечных координат уже рассматривались нами в главе 2.

Структура функции `PlutoPos` отличается от остальных. Координаты сначала вычисляются относительно фиксированной эклиптики 1950,0, а затем преобразуются к равноденствию текущей даты. Такой метод приходится использовать из-за большого наклонения орбиты Плутона. Полное разложение координат в ряды привело бы к появлению большого количества вековых членов. Также следует заметить, что `PlutoPos` можно использовать только в диапазоне с 1890 по 2100 год. Дело в том, что ряды для Плутона были получены не из теории возмущений, а путем Фурье-анализа численно проинтегрированных эфемерид, покрывающих этот интервал времени. За пределами указанного диапазона ошибка начинает очень быстро расти и уже через несколько лет становится более половины градуса.

Для простоты использования вызовы подпрограмм собраны в функцию `PertPosition`, которая по аналогии с рассмотренной нами раньше `KepPosition` позволяет вычислить гелиоцентрические координаты любой планеты.

```
//-------------------------------------------------------------------------
// PertPosition: Вычисляет положения планет, по аналитическим рядам
//   Planet    Идентифицирует планету
//   T         Время в юлианских столетиях от эпохи J2000
// <return>:   Гелиоцентрическое положение [а.е.]
//             относительно эклиптики и равноденствия даты
//-------------------------------------------------------------------------
Vec3D PertPosition (PlanetType Planet, double T)
{
  Vec3D r; // Нуль-вектор
  switch ( Planet )
  {
    case Sun:       break;
    case Mercury:  r = MercuryPos(T);  break;
    case Venus:    r = VenusPos(T);    break;
    case Earth:    r = (-SunPos(T));   break;
    case Mars:     r = MarsPos(T);     break;
    case Jupiter:  r = JupiterPos(T);  break;
    case Saturn:   r = SaturnPos(T);   break;
    case Uranus:   r = UranusPos(T);   break;
    case Neptune:  r = NeptunePos(T);  break;
    case Pluto:    r = PlutoPos(T);
  }
  return r;
};
```

Выбор планеты обеспечивается заданием параметра Planet, принадлежащего типу PlanetType:

```
enum PlanetType { Sun, Mercury, Venus, Earth, Mars,
                  Jupiter, Saturn, Uranus, Neptune, Pluto },
```

Значениям этого типа соответствуют числа от 0 до 9, что позволяет перебирать планеты в цикле.

Для XIX и XX веков программа обеспечивает точность до нескольких секунд дуги (табл. 6.2). Для более отдаленных моментов времени следует ожидать меньшей точности.

Таблица 6.2. Средние ошибки функций MercuryPos...PlutoPos на интервале времени 1750–2250 гг. (1890–2100 гг. для Плутона)

Планета	Δl [″]	Δb [″]	Δr [10^{-6} а.е.]
Меркурий	1,0–1,5	0,7–1,0	1,0– 1,5
Венера	0,5–1,0	0,2–2,5	0,5– 1,5
Солнце, Земля	0,5–2,0	0,0–0,1	0,4– 1,5
Марс	0,5–2,5	0,1–1,0	3 – 10
Юпитер	2– 8	0,7–1,0	20 – 30
Сатурн	2– 11	0,8–1,5	40 – 100
Уран	3– 8	0,7–1,0	50 – 200
Нептун	3– 40	1,0–2,0	500 –1000
Плутон	1– 9	0,2–2,2	200 –1000

Во-первых, положенные в основу теории были разработаны в конце XIX столетия и потому могли основываться только на тех наблюдениях, которые были

доступны в то время. Во-вторых, в программе игнорируются возмущающие члены, зависящие от T^3. Текущие значения этих членов очень малы (<0,1″), однако на больших временных масштабах они вполне могут увеличиться в тысячи раз.

6.4. Видимые и астрометрические координаты

Пользуясь эфемеридами планет, опубликованными в хорошем ежегоднике, вы обязательно обнаружите примечание, поясняющее, являются ли приводимые значения координат *геометрическими*, *видимыми* или *астрометрическими*. Различием между этими тремя видами координат можно пренебречь, если вас устраивает точность около 1 минуты дуги или даже немного хуже. Однако для того, чтобы в полной мере использовать точность, которую обеспечивают программы, рассмотренные в предыдущем разделе, необходимо знать и понимать наиболее важные поправки, применяемые к координатам планет.

В то время как геоцентрические координаты указывают точку пространства, в которой планета *располагается* в указанный момент времени t, видимые и астрометрические координаты задают направление, в котором планета *наблюдается*.

Геометрические координаты обычно используются только для задания гелиоцентрических положений планет. При этом используемая система координат однозначно определяется заданным равноденствием. Набор функций `MercuryPos`, ..., `PlutoPos` возвращает *геометрические гелиоцентрические эклиптические координаты, отнесенные к равноденствию текущей даты (то есть к эпохе, для которой производятся вычисления)*. Это утверждение, естественно, не относится к функции `SunPos`, которая вычисляет соответствующее *геоцентрическое* положение Солнца.

В противоположность этому при задании геоцентрических положений (кроме расстояний) обычно приводят астрометрические или видимые координаты. Причина состоит в том, что вследствие своего движения и световой задержки планеты не наблюдаются в тех положениях, которые они занимают на момент расчета. Мы уже упоминали об этом эффекте, когда обсуждали орбиты комет. Поскольку длина пути, проходимого светом, недоступна прямому наблюдению, геоцентрические эфемериды обычно содержат геометрическое значение расстояния.

6.4.1. Аберрация и световая задержка

Астрометрические координаты задают направление, по которому на Землю приходит луч света наблюдаемой планеты. Этот луч света связывает положение Земли в момент наблюдения t с положением планеты в момент t', когда этот свет был ею испущен. Световая задержка $\tau = t - t'$ примерно соответствует геометрическому расстоянию

$$\Delta_0 = |\boldsymbol{r}_0|, \quad \text{где} \quad \boldsymbol{r}_0 = \boldsymbol{r}_\odot(t) + \boldsymbol{r}_p(t), \tag{6.11}$$

то есть $\tau \approx \Delta_0 / c$ (смысл обозначений проиллюстрирован на рис. 6.2). За время световой задержки движение планет можно считать прямолинейным. Поэтому астрометрические координаты планеты в момент наблюдения t можно вычислить так:

$$\boldsymbol{r} = \boldsymbol{r}_{\odot}(t) + \boldsymbol{r}_P(t') \approx \boldsymbol{r}_{\odot}(t) + \left\{ \boldsymbol{r}_p(t) - \boldsymbol{v}_p(t) \cdot \frac{\Delta_0}{c} \right\}. \tag{6.12}$$

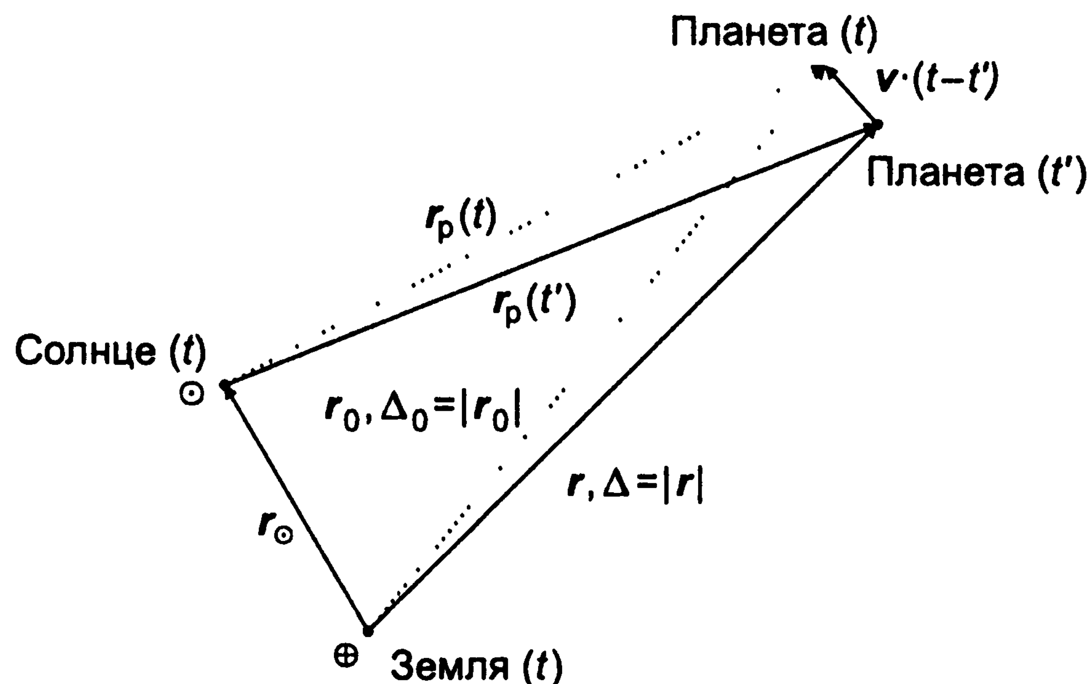


Рис. 6.2. Перемещение планеты за время световой задержки

Разница между *астрометрическими* координатами $\boldsymbol{r}$ и геометрическими координатами $\boldsymbol{r}_0$ называется *световой поправкой*. По астрометрическим координатам положение небесного тела можно наносить на звездную карту и сравнивать с известными положениями звезд на фотопластинках.

Однако если при наблюдении с Земли мы воспользуемся для наведения телескопа астрометрическими координатами, то обнаружим, что наблюдаемое положение планеты отличается от расчетного. Различие будет составлять около 20 угловых секунд. Эта ошибка возникает вследствие того, что астрометрические координаты определяют направление падающего на Землю луча света в системе отсчета, связанной с Солнцем. В то же время наблюдатель участвует в суточном вращении Земли вокруг своей оси и в ее годичном движении вокруг Солнца. Вследствие этих движений и конечности скорости света меняется видимое направление, по которому свет от планеты приходит на Землю. Для иллюстрации этого явления часто приводят пример дождя, наблюдаемого из салона автомобиля. Хотя относительно земли капли падают вертикально, наблюдателю в автомобиле кажется, что они двигаются под углом к горизонту, величина которого зависит от скорости и направления движения машины.

Поправка, учитывающая переход от неподвижной системы координат к движущейся, в равной мере влияет на координаты не только планет, но и неподвижных звезд. Таким образом, речь идет о *звездной* аберрации. В каталогах и в звездных атласах приводятся положения звезд, как они видны неподвижному наблюдателю в Солнечной системе, то есть непосредственно они сопоставимы именно с астрометрическими координатами планет. С другой стороны, земной наблюдатель для наведения инструмента должен использовать видимые положения звезд, которые подвержены периодическим вариациям на протяжении года.

Исчерпывающее описание явления звездной аберрации дает специальная теория относительности, но его изложение увело бы нас слишком далеко. Поскольку скорость Земли мала по сравнению со скоростью света, мы можем ограничиться полуклассическим описанием, которое в пределах требуемой точности приводит к тем же результатам, что и строгая теория.

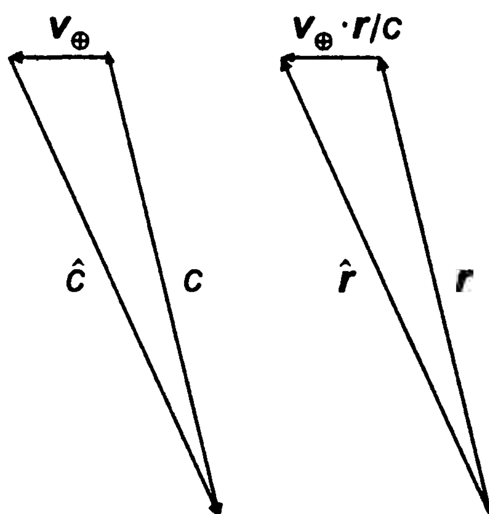


Рис. 6.3. Эффект звездной аберрации

Пусть $\boldsymbol{r}$ — вектор астрометрических координат планеты. Тогда падающий на Землю световой луч можно описать вектором скорости света (рис. 6.3).

$$\boldsymbol{c} = -c \cdot \frac{\boldsymbol{r}}{|\boldsymbol{r}|}$$

Вычтем из этого вектора скорость $v_\oplus$ Земли относительно Солнца, чтобы получить вектор $\hat{\boldsymbol{c}}$, приведенный к системе координат наблюдателя, находящегося на Земле:

$$\hat{\boldsymbol{c}} = \boldsymbol{c} - \boldsymbol{v}_\oplus.$$

Таким образом, наблюдатель увидит планету в направлении

$$\hat{\boldsymbol{r}} = -|\boldsymbol{r}| \cdot \frac{\hat{\boldsymbol{c}}}{c} = \boldsymbol{r} + \boldsymbol{v}_\oplus \cdot \frac{|\boldsymbol{r}|}{c} \approx \boldsymbol{r} + \boldsymbol{v}_\oplus \cdot \frac{\Delta_0}{c}. \quad (6.13)$$

Чтобы полученные таким образом координаты можно было назвать ***видимым*** положением планеты, их необходимо представить в системе координат, одна из осей которой в точности совпадает с текущим положением оси вращения Земли. Для этого требуется не только выполнить приведение к равноденствию даты, но также учесть нутацию.

Нутации будет посвящен следующий раздел, но прежде, чем перейти к нему, покажем, как вычислить гелиоцентрический вектор скорости планеты, необходимый для учета эффекта аберрации. Поскольку скорость планеты v мала по сравнению со скоростью света c, поправки за световую задержку и звездную аберрацию имеют величину порядка одной угловой минуты. Поэтому для того, чтобы учесть влияние аберрации с точностью до одной угловой секунды, нам достаточно знать скорость планеты с точностью до двух десятичных цифр. Благодаря этому для вычисления скорости вместо сложных рядов можно воспользоваться приближением кеплеровых орбит. Функция `KepVelocity` вычисляет гелиоцентри-

ческий вектор скорости планеты. В ее основе лежит тот же принцип, что и в функции KepPosition, которую мы рассматривали в разделе 5.2.

```
//------------------------------------------------------------------
// KepVelocity: Вычисляет скорость планеты по элементам кеплеровой орбиты
//   Planet    Идентифицирует планету
//   T         Время в юлианских столетиях от эпохи J2000
// <return>:   Гелиоцентрическая скорость [а.е./сут.],
//             относительно эклиптики и равноденствия даты
//------------------------------------------------------------------
Vec3D KepVelocity (PlanetType Planet, double T)
{
  Vec3D r, v;
  State (Planet, T, r, v);  return v;
}
```

Элементы орбит, занесенные в функцию State, точны на начало 2000 года, однако они обеспечивают необходимую точность на пару лет до и после этого момента.

6.4.2. Нутация

Видимые координаты — это координаты, предназначенные для использования при наведении телескопа на объект. Поскольку полярная ось монтировки телескопа всегда строго параллельна оси вращения Земли, используемая система координат должна иметь такую же ориентацию. В первую очередь это означает, что видимые координаты должны быть приведены к равноденствию даты, то есть к текущему положению экватора, эклиптики и точки весеннего равноденствия. Однако кроме прецессии существует еще один эффект, влияющий на положение земной оси, который необходимо учитывать. В то время как прецессия описывает вековое, то есть долгосрочное изменение положения земной оси, *нутация* представляет собой дополнительные небольшие периодические колебания ее положения. В результате нутации *истинный* полюс Земли каждые 18,6 года совершает оборот вокруг *среднего* полюса, положение которого описывается прецессией. Период нутации определяется периодом вращения линии узлов, то есть временем, за которое ее восходящий узел ☊ лунной орбиты совершает один полный оборот. Нутация в долготе ($\Delta\psi$), то есть разность между истинным и средним весенним равноденствием, достигает по амплитуде 17 угловых секунд. Наклон эклиптики при этом варьируется на величину $\Delta\varepsilon \approx 9''$ (рис. 6.4).

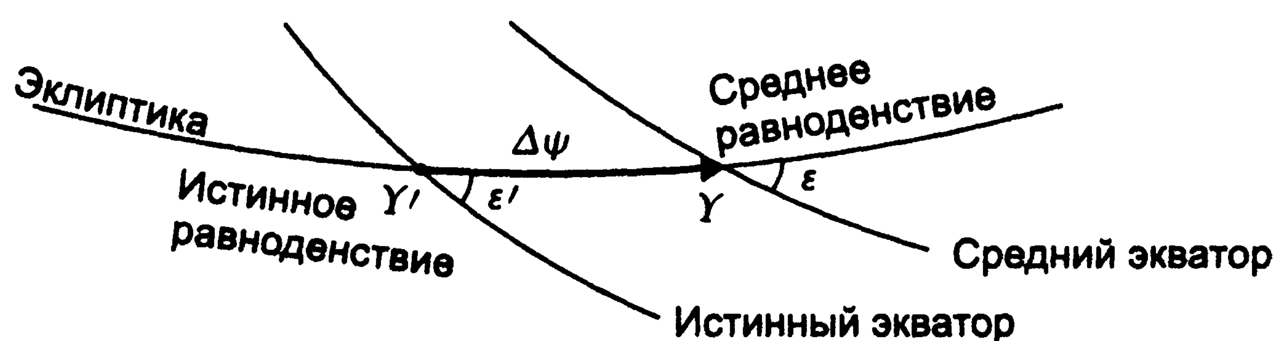


Рис. 6.4. Изменение взаимного расположения экватора, эклиптики и точки весеннего равноденствия под действием нутации

Обе эти величины складываются примерно из сотни отдельных членов. Тем не менее для большинства задач достаточно учитывать всего несколько:

$$\begin{aligned}\Delta\psi &= -\,17''\!.200\cdot\sin\Omega + 0''\!.206\cdot\sin(2\Omega) + 0''\!.143\cdot\sin(l') -\\ &\quad -\,1''\!.319\cdot\sin(2(F-D+\Omega)) - 0''\!.227\cdot\sin(2(F+\Omega)),\\ \Delta\varepsilon &= +\,9''\!.203\cdot\cos\Omega - 0''\!.090\cdot\cos(2\Omega) +\\ &\quad +\,0''\!.574\cdot\cos(2(F-D+\Omega)) + 0''\!.098\cdot\cos(2(F+\Omega)).\end{aligned} \tag{6.14}$$

Помимо долготы восходящего узла Ω эти члены зависят также от ряда других аргументов (F, D, l'), являющихся функциями средних долгот и аномалий Солнца и Луны (см. (8.2—8.5)).

Истинные экваториальные координаты $\boldsymbol{r}' = (x', y', z')$ получаются из средних координат $\boldsymbol{r} = (x, y, z)$ умножением на матрицу нутации

$$\boldsymbol{N} = \boldsymbol{R}_x(-\varepsilon-\Delta\varepsilon)\,\boldsymbol{R}_z(\Delta\psi)\boldsymbol{R}_x(\varepsilon) \tag{6.15}$$

с компонентами

$$\begin{aligned}n_{11} &= +\cos(\Delta\psi)\\ n_{21} &= +\cos(\varepsilon')\cdot\sin(\Delta\psi)\\ n_{31} &= +\sin(\varepsilon')\cdot\sin(\Delta\psi)\\ \\ n_{12} &= -\cos(\varepsilon)\cdot\sin(\Delta\psi)\\ n_{22} &= +\cos(\varepsilon)\cdot\cos(\varepsilon')\cdot\cos(\Delta\psi)+\sin(\varepsilon)\cdot(\varepsilon')\\ n_{32} &= +\cos(\varepsilon)\cdot\sin(\varepsilon')\cdot\cos(\Delta\psi)-\sin(\varepsilon)\cdot\cos(\varepsilon')\\ \\ n_{13} &= -\sin(\varepsilon)\cdot\sin(\Delta\psi)\\ n_{23} &= +\sin(\varepsilon)\cdot\sin(\varepsilon')\cdot\cos(\Delta\psi)-\cos(\varepsilon)\cdot\sin(\varepsilon')\\ n_{33} &= +\sin(\varepsilon)\cdot\sin(\varepsilon')\cdot\cos(\Delta\psi)+\cos(\varepsilon)\cdot\cos(\varepsilon')\end{aligned} \tag{6.16}$$

которая заключает в себе три базовых поворота (см. рис. 6.4). Здесь ε обозначает средний (2.9), а $\varepsilon' = \varepsilon + \Delta\varepsilon$ — истинный наклон эклиптики.

Функция `NutMatrix` принимает в качестве параметра время T, выраженное в юлианских столетиях, прошедших с эпохи J2000, и вычисляет параметры преобразования от средних координат к истинным, используя приведенные упрощенные формулы для углов нутации.

```
//------------------------------------------------------------------------
//
// NutMatrix: Преобразование от средних к истинным экватору и равноденствию
//   T          Время в юлианских столетиях от эпохи J2000
// <return>:   матрица нутации
//
//------------------------------------------------------------------------
Mat3D NutMatrix (double T)
{
  // Переменные
```

```
  double  ls, D, F, N;
  double  eps, dpsi, deps;
  // Средние аргументы лунных и солнечных движений
  ls = pi2*Frac(0 993133+  99 997306*T);  // средняя аномалия Солнца
  D  = pi2*Frac(0 827362+1236.853087*T),  // разность долгот Луна-Солнце
  F  = pi2*Frac(0 259089+1342.227826*T);  // средний аргумент широты
  N  = pi2*Frac(0.347346-   5 372447*T);  // долгота восходящего узла
  // Углы нутации
  dpsi = ( -17.200*sin(N)   - 1.319*sin(2*(F-D+N)) - 0 227*sin(2*(F+N))
         + 0 206*sin(2*N) + 0.143*sin(ls) ) / Arcs;
  deps = ( + 9.203*cos(N)   + 0.574*cos(2*(F-D+N)) + 0 098*cos(2*(F+N))
         - 0.090*cos(2*N)                  ) / Arcs;
  // Средний наклон эклиптики
  eps  = 0 4090928-2.2696E-4*T;
  // Матрица нутации
  return  R_x(-eps-deps)*R_z(-dpsi)*R_x(+eps);
}
```

6.5. Программа Planpos

Программа Planpos вычисляет положение Солнца и девяти больших планет на заданную дату. Ввиду ограничений функции PlutoPos координаты Плутона определяются только в диапазоне между 1890 и 2100 годами. Можно задать эпоху, к которой будут приведены положения планет. Имеются три возможности: вычислить видимые координаты, отнесенные к истинному равноденствию выбранной даты, или астрометрические координаты на одну из двух наиболее часто используемых эпох — B1950 и J2000. Видимые координаты всегда содержат поправки за прецессию, нутацию, аберрацию и световую задержку. Их можно использовать, например,для наведения на объект телескопа, установленного на экваториальной монтировке. Астрометрические координаты требуются для нанесения планет на звездную карту, составленную для соответствующей эпохи. В них учтены только прецессия и световая задержка.

```
//------------------------------------------------------------------------
// File:    Planpos cpp
// Purpose. Эфемериды больших планет
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//------------------------------------------------------------------------

#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>
#include "APC_Const.h"
#include "APC_Kepler h"
#include "APC_Math h"
#include "APC_Planets h"
#include "APC_PrecNut h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time h"
#include "APC_VecMat3D h"
```

```
using namespace std:

//------------------------------------------------------------------------------
// Основная программа
//------------------------------------------------------------------------------
void main()
{
  // Названия планет
  const char* Name[] =
    { "Sun". "Mercury". "Venus". "Earth". "Mars".
      "Jupiter". "Saturn". "Uranus". "Neptune". "Pluto" }:
  // Переменные
  PlanetType Planet.
  char       Mode:
  int        iPlanet. last:
  int        year. month. day:
  double     Hour. MJD. T:
  double     dist. fac:
  Vec3D      R_Sun. r_helioc. r_geoc. r_equ.
  Mat3D      P:
  bool       End = false.
  // Заголовок
  cout << endl
       << "   PLANPOS· geocentric and heliocentric planetary positions " << endl
       << "            (c) 1999 Oliver Montenbruck. Thomas Pfleger     " << endl
       << endl:

  // Цикл обработки команд
  do {
    // Ввод команды
    cout << endl
         << " (J) J2000 astrometric      (B) B1950 astrometric  " << endl
         << " (A) apparent coordinates   (E) exit               " << endl
         << endl
         << " Enter option ... ":
    cin >> Mode: cin ignore(81.'\n'): Mode = tolower(Mode).   cout << endl:

    // Завершение. если Mode=Exit
    if (Mode=='e') { End=true: break: }
    // Повтор. при ошибочном значении Mode
    if (Mode!='j' && Mode!='b' && Mode!='a') continue:

    // Эпоха
    cout << " Date (yyyy mm dd hh.hhh) ..  ".
    cin >> year >> month >> day >> Hour: cin.ignore(81.'\n').
    cout << endl << endl << endl:
    MJD = Mjd(year.month.day)+Hour/24.0.
    T   = ( MJD - MJD_J2000 ) / 36525 0.

    // Шапка таблицы
    cout << " Date:  " << DateTime(MJD.HHh) << " (ET)"
         << "     JD· " << (MJD+2400000.5)
         << setw(18) << "equinox ":
    switch (Mode) {
      case 'a': cout << "of date" << endl: break:
      case 'j': cout << "J2000"   << endl: break:
      case 'b': cout << "B1950"   << endl:
```

```
}
cout << endl
     << endl
     << "               l            b          r          "
     << "RA          Dec        delta"
     << endl
     << "             o  '  \"     o  '  \"       AU         "
     << "h  m  s      o  '  \"      AU"
     << endl;

// Эклиптические координаты Солнца: равноденствие даты
R_Sun = SunPos(T);
// Цикл по планетам
if ( (-1.1<T) && (T<1.0) ) last=Pluto; else last=Neptune;
for (iPlanet=Sun; iPlanet<=last; iPlanet++) {
  // Гелиоцентрические эклиптические координаты планет: равноденствие даты
  Planet = (PlanetType) iPlanet;
  r_helioc = PertPosition(Planet, T);
  // Геоцентрические эклиптические координаты: равноденствие даты
  r_geoc = r_helioc + R_Sun;
  // Поправка за световую задержку/аберрацию первого порядка
  dist = Norm(r_geoc);
  fac = dist/c_light;
  if (Mode=='a')
    r_geoc -= fac*(KepVelocity(Planet,T)-KepVelocity(Earth,T));
  else
    r_geoc -= fac*KepVelocity(Planet,T);
  // Прецессия и экваториальные координаты
  switch (Mode) {
    case 'a':  r_equ    = NutMatrix(T) * Ecl2EquMatrix(T) * r_geoc;
               break;
    case 'j':  P        = PrecMatrix_Ecl(T,T_J2000);
               r_helioc = P * r_helioc;
               r_geoc   = P * r_geoc;
               r_equ    = Ecl2EquMatrix(T_J2000) * r_geoc;
               break;
    case 'b':  P        = PrecMatrix_Ecl(T,T_B1950);
               r_helioc = P * r_helioc;
               r_geoc   = P * r_geoc;
               r_equ    = Ecl2EquMatrix(T_B1950) * r_geoc;
  }

  // Вывод
  cout << " " << setw(8) << left << Name[iPlanet] << right;
  cout << fixed << setprecision(1)
       << setw(11) << Angle(Deg*r_helioc[phi],DMMSSs) << " "
       << setw(11) << Angle(Deg*r_helioc[theta],DMMSSs);
  if (Planet<=Earth)
    cout << setprecision(6) << setw(11) << r_helioc[r];
  else
    cout << setprecision(5) << setw(10) << r_helioc[r] << " ";
  cout << setprecision(2) << setw(13) << Angle(Deg*r_equ[phi]/15.0,DMMSSs)
       << " " << showpos
       << setprecision(1) << setw(11) << Angle(Deg*r_equ[theta],DMMSSs)
       << noshowpos;
  if (Planet<=Earth)
    cout << setprecision(6) << setw(11) << dist;
```

```
      else
        cout << setprecision(5) << setw(10) << dist << " ";
      cout << endl;
    };

    // Пояснения
    cout << endl
         << " l,b,r:   heliocentric ecliptic (geometric) " << endl
         << " RA,Dec:  geocentric equatorial ";
    if (Mode=='a')
      cout << "(apparent)" << endl;
    else
      cout << "(astrometric)" << endl;
    cout << " delta:   geocentric distance   (geometric)" << endl
         << endl;
  }
  while (!End);
}
```

Сразу после запуска программа Planpos запрашивает эпоху, к которой должны быть отнесены вычисляемые положения планет. Можно ввести A (apparent — видимые координаты, равноденствие даты), J (J2000, астрометрические координаты) или B (B1950, астрометрические координаты). Выбранный вариант всегда выводится на экран для проверки, но не сразу, а только после вычисления координат. Для завершения работы программы следует ввести команду E.

В приводимом далее примере мы выбираем видимые координаты. В ответ на запрос вводим точный момент времени, на который нам требуется вычислить координаты. Обратите внимание, что его следует задавать по эфемеридному времени (ET)! В качестве примера мы задали 0 часов 1 января 1989 года. (Как обычно, все данные, вводимые пользователем, выделены курсивом.) После этого Planpos выдает требуемые координаты.

```
   PLANPOS: geocentric and heliocentric planetary positions
            (c) 1999 Oliver Montenbruck, Thomas Pfleger

 (J) J2000 astrometric        (B) B1950 astrometric
 (A) apparent coordinates     (E) exit

 Enter option ... a

 Date (yyyy mm dd hh.hhh) ... 1989 01 01 0.0

Date:  1989/01/01 00.0 (ET)     JD· 2447527.5          equinox of date

                l             b           r           RA          Dec          delta
            o  '  "      o  '  "        AU         h  m  s      o  '  "         AU
 Sun       0 00 00.0    0 00 00.0   0.000000   18 45 53.66 -23 01 25.6    0.983309
 Mercury 347 56 35.7 -  6 05 21.5   0.370100   19 59 16.60 -22 34 12.1    1 175632
 Venus   226 05 15.0    1 43 26.9   0.723666   17 07 15.21 -22 03 57.5    1.522289
 Earth   100 33 09.4    0 00 00.2   0.983309    0 00 00.00 + 0 00 00.0    0.000000
 Mars     60 32 51.9    0 21 19.3   1 49745     1 13 47.44 + 8 24 05.2    0.97649
 Jupiter  64 30 07.8 -  0 45 53.2   5.03195     3 38 35.14 +18 33 07.5    4.27637
 Saturn  275 06 53.5    0 47 14.5  10.04354    18 24 14.50 -22 36 28.2   11.02274
 Uranus  271 18 42.6 -  0 13 47.8  19.31483    18 07 39.22 -23 39 00.8   20.28599
```

```
Neptune 279 54 35 7    0 55 55.8  30.21917   18 42 54.09 -22 10 16 6  31.20230
Pluto   222 55 15.8   15 52 34.4  29.65861   15 06 35.06 - 1 16 18.1  30 17673

l.b.r·   heliocentric ecliptic (geometric)
RA.Dec·  geocentric equatorial (apparent)
delta    geocentric distance   (geometric)
```

Заметьте, что гелиоцентрические эклиптические координаты Солнца и геоцентрические экваториальные координаты Земли всегда равны нулю.

Для сравнения вычислим положения планет на тот же момент времени, но в координатах эпохи J2000. Небольшие различия в полученных результатах связаны с прецессией за 11 лет, разделяющих эту эпоху и 1989 год.

```
(J) J2000 astrometric       (B) B1950 astrometric
(A) apparent coordinates    (E) exit

Enter option . . j

Date (yyyy mm dd hh hhh) ..  1989 01 01 0.0
Date:  1989/01/01 00 0 (ET)     JD  2447527.5         equinox J2000

              l            b            r          RA           Dec         delta
          o   '   "    o   '   "       AU        h  m  s      o   '   "      AU
Sun       0 00 00.0    0 00 00.0   0.000000  18 46 34 57 -23 00 32 4   0.983309
Mercury 348 05 49.4 -  6 05 22.1   0 370100  19 59 56.54 -22 32 12 4   1 175632
Venus   226 14 28.3    1 43 22.9   0.723666  17 07 55 78 -22 04 40.7   1.522289
Earth   100 42 22.5    0 00 05 2   0 983309   0 00 00.00 + 0 00 00 0   0.000000
Mars     60 42 05.0    0 21 24.1   1.49745    1 14 21.40 + 8 27 27 7   0 97649
Jupiter  64 39 21 0 -  0 45 48 3   5.03195    3 39 11 61 +18 35 03 6   4.27637
Saturn  275 16 06.7    0 47 09.5  10 04354   18 24 55 40 -22 35 56.1  11.02274
Uranus  271 27 55.8 -  0 13 52.9  19.31483   18 08 20 48 -23 38 44 9  20 28599
Neptune 280 03 48.9    0 55 50 8  30 21917   18 43 34 77 -22 09 26 4  31 20230
Pluto   223 04 29.9   15 52 30 6  29 65861   15 07 09 55 - 1 18 40 5  30.17673

l.b.r·   heliocentric ecliptic (geometric)
RA.Dec.  geocentric equatorial (astrometric)
delta·   geocentric distance   (geometric)
```

Как видим, различия очень малы, но пренебрегать ими все-таки нельзя. Сравнивая эти результаты с эфемеридами, публикуемыми в ежегоднике, обратите внимание, что нередко наряду с видимыми координатами всех остальных планет для Плутона приводят астрометрические координаты. Так поступают, потому что эту очень слабую планету обычно удобнее отыскивать не непосредственно по видимым координатам, а по положению относительно соседних звезд.

```
(J) J2000 astrometric       (B) B1950 astrometric
(A) apparent coordinates    (E) exit

Enter option .   e
```

Вводя команду E, мы завершаем работу программы.

7. Физические эфемериды планет

В предыдущей главе мы рассмотрели методы точного вычисления положений больших планет Солнечной системы. Однако каждому, кто регулярно наблюдает планеты, Солнце или Луну важно знать, как будет выглядеть наблюдаемое небесное тело. Важнейшее значение имеют видимая звездная величина и угловой диаметр. Тем, кто собирается наблюдать и регистрировать детали на поверхности или атмосферные образования, необходимо знать ориентацию оси вращения планеты (то есть планетографической системы координат) относительно луча зрения, а также условия освещения на планете. В этой главе мы рассмотрим, как определить эти параметры. При этом мы постараемся достичь точности, приемлемой для большинства наблюдательных задач.

Необходимые для вычислений данные о размерах, форме и вращении планет взяты нами из [33] (см. раздел «Литература» в конце книги). Вычисляемые по этим данным физические эфемериды согласуются со значениями, публикуемыми в «Astronomical Almanac».

Поскольку физические эфемериды зависят от относительного расположения Солнца, Земли и планеты, нам потребуются гелиоцентрические координаты Земли и планет. Для их определения мы будем пользоваться функцией `PertPosition`, которая была описана в главе 6 и позволяет вычислять точные координаты планет на большом интервале времени.

7.1. Вращение

Ориентация Земли в пространстве задается положением северного полюса мира и звездным временем. Точно так же ориентация любой другой планеты описывается положением соответствующего северного полюса и углом поворота вокруг оси вращения. В соответствии с рекомендациями Международного астрономического союза (МАС) северным полюсом планеты, астероида или спутника по определению считается полюс, находящийся в северном полушарии относительно *неизменной плоскости* Солнечной системы. Эта плоскость перпендикулярна совокупному вектору углового момента Солнечной системы, однако упрощенно ее можно представлять себе как среднюю плоскость планетных орбит. Поскольку наклонение планетных орбит к плоскости орбиты Земли невелико, неизменная плоскость почти в точности совпадает с эклиптикой.

В соответствии с приведенным определением не у всех планет направление вращения совпадает с направлением вращения Земли. Венера, Уран и Плутон, если смотреть со стороны северного полюса, вращаются по часовой стрелке. В отличие от *прямого направления вращения*, их вращение называют *обратным*.

Таблица 7.1. Ориентация осей вращения Солнца и планет (МАС, 1994)

	α_0 (J2000)	δ_0 (J2000)	
Солнце	$286^\circ\!.13$	$+63^\circ\!.87$	
Меркурий	$281^\circ\!.01 - 0^\circ\!.033T$	$+61^\circ\!.45 - 0^\circ\!.005T$	
Венера	$272^\circ\!.76$	$+67^\circ\!.16$	
Земля	$0^\circ\!.00 - 0^\circ\!.641T$	$+90^\circ\!.00 - 0^\circ\!.557T$	
Марс	$317^\circ\!.681 - 0^\circ\!.108T$	$+52^\circ\!.886 - 0^\circ\!.06T$	
Юпитер	$268^\circ\!.05 - 0^\circ\!.009T$	$+64^\circ\!.49 + 0^\circ\!.003T$	
Сатурн	$40^\circ\!.589 - 0^\circ\!.036T$	$+83^\circ\!.537 - 0^\circ\!.004T$	
Уран	$257^\circ\!.311$	$-15^\circ\!.175$	
Нептун	$299^\circ\!.36 + 0^\circ\!.70 \sin N$	$+43^\circ\!.46 - 0^\circ\!.51 \cos N$	$N = 357^\circ\!.85 + 52^\circ\!.316T$
Плутон	$313^\circ\!.02$	$+9^\circ\!.09$	

Важно уметь определять ориентацию не только для планет, но и для Солнца. Описываемая далее вычислительная процедура вполне применима и к нему, а некоторые особенности ее использования обсуждаются в отдельном разделе.

В табл. 7.1 содержатся данные, описывающие ориентацию осей вращения Солнца и планет. В колонках приводятся значения прямого восхождения α_0 и склонения δ_0 северного полюса, отнесенные к эпохе J2000. Аргумент

$$T = (\mathrm{JD} - 2451545)/36525, \tag{7.1}$$

как обычно, представляет число юлианских столетий, прошедших с эпохи J2000.

7.1.1. Позиционный угол оси вращения

Положение оси вращения планеты, как ее видит наблюдатель, обычно описывается позиционным углом и широтой Земли относительно экватора планеты (рис. 7.1). Позиционный угол ϑ — это угол между проекцией оси вращения планеты на небесную сферу и направлением на северный полюс мира наблюдателя. Позиционный угол отсчитывается от севера ($\vartheta = 0°$) к востоку ($\vartheta = 90°$).

Вычисление позиционного угла мы начнем с определения геоцентрических экваториальных координат планеты $\boldsymbol{r} = (x, y, z)$. Мы будем также использовать вектор

$$\boldsymbol{d} = \begin{pmatrix} d_x \\ d_y \\ d_z \end{pmatrix} = \begin{pmatrix} \cos\alpha_0 \cos\delta_0 \\ \sin\alpha_0 \cos\delta_0 \\ \sin\delta_0 \end{pmatrix} \tag{7.2}$$

отложенный от центра планеты и задающий направление ее оси. На основе вектора $\boldsymbol{r}$ построим три взаимно перпендикулярных единичных вектора $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$. Вектор $\boldsymbol{e}_1$ направим к планете, $\boldsymbol{e}_2$ — в направлении возрастания прямых восхождений (то есть к востоку), а вектор $\boldsymbol{e}_3 = \boldsymbol{e}_1 \times \boldsymbol{e}_2$ — в сторону увеличения склонений (к северу), как показано на рис. 7.2. По этому построению получаем:

$$e_1 = \frac{1}{r}\begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad e_2 = \frac{1}{\rho}\begin{pmatrix} -y \\ +x \\ 0 \end{pmatrix}, \quad e_3 = \frac{1}{r\rho}\begin{pmatrix} -xz \\ -yz \\ x^2 + y^2 \end{pmatrix}, \tag{7.3}$$

где

$$r = \sqrt{x_2 + y_2 + z_2} \quad \text{и} \quad \rho = \sqrt{x_2 + y_2}$$

соответственно длина радиус-вектора и его проекция на плоскость экватора. Позиционный угол ϑ, под которым мы видим направляющий вектор $\boldsymbol{d}$, измеряется в плоскости, задаваемой векторами e_2 и e_3, которая перпендикулярна лучу зрения. Проектируя вектор $\boldsymbol{d}$ на e_2 и e_3, мы получаем следующие выражения для определения позиционного угла ϑ:

$$\begin{aligned} \cos\vartheta &= e_3 \cdot \boldsymbol{d} = ((-xz)d_x + (-yz)d_y + (x^2 + y^2)d_z)/(r\rho), \\ \sin\vartheta &= e_2 \cdot \boldsymbol{d} = ((-y)d_x + (x)d_y)/\rho. \end{aligned} \tag{7.4}$$

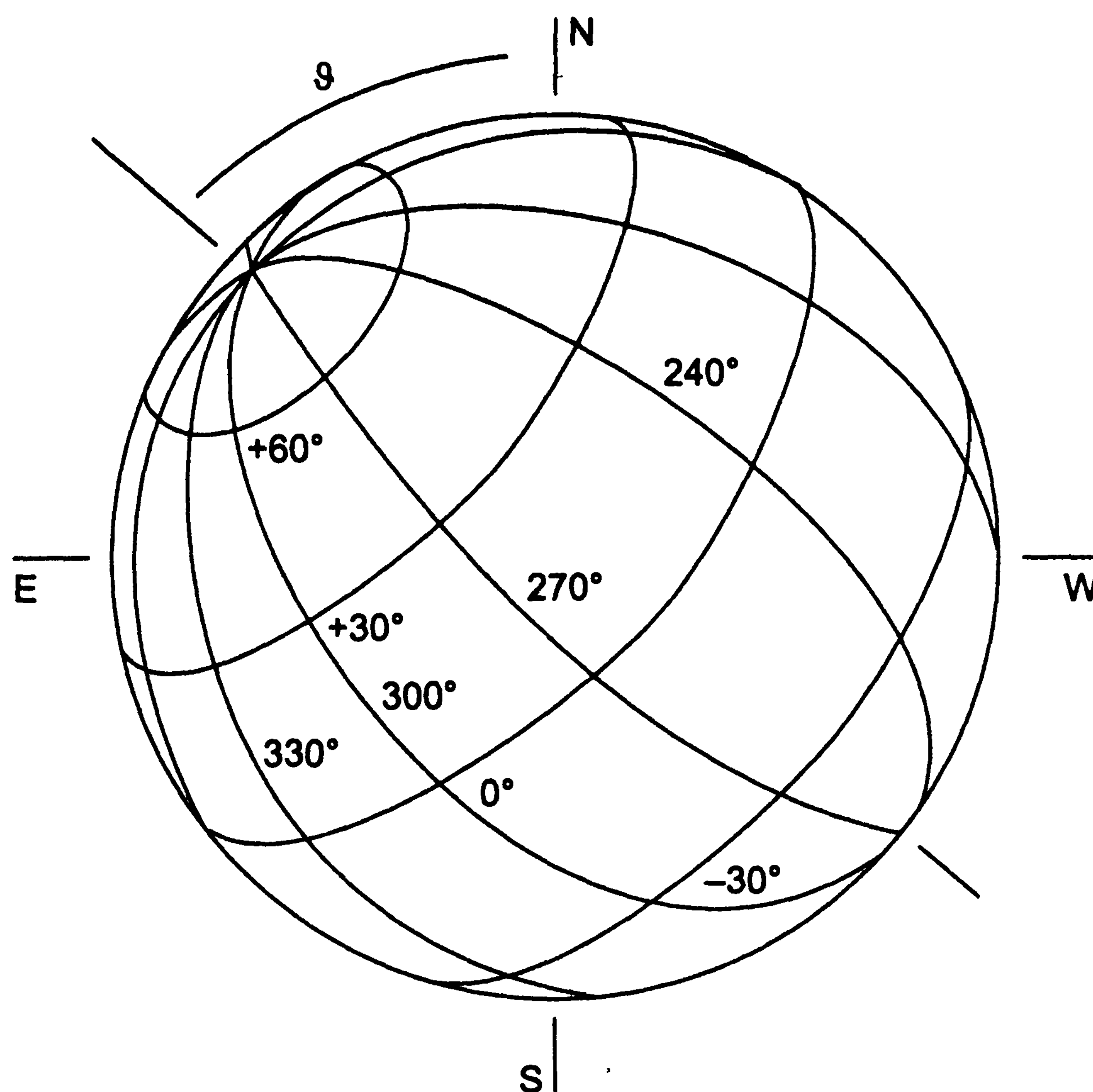


Рис. 7.1. Позиционный угол оси вращения и планетографические координаты (на примере Марса в 0^h 1 октября 1993 года ET; $\vartheta = 37{,}^\circ9$, $\lambda_\oplus = 276{,}^\circ1$, $\varphi_\oplus = 20{,}^\circ4$)

Функция PosAng использует эти уравнения для вычисления позиционного угла.

```
//----------------------------------------------------------------------------
// PosAng: Вычисляет позиционный угол по отношению к лучу зрения r
//   r          Положение наблюдаемого тела
//   d          Направление, для которого вычисляется позиционный угол
// <return>:    Позиционный угол в [рад]; диапазон [0, 2*pi]
// Примечание: r и d должны относиться к одной системе координат и эпохе
```

```
//------------------------------------------------------------------------
double PosAng (const Vec3D& r, const Vec3D& d)
{
  Vec3D  e_1, e_2, e_3;
  double  c, s, phi;
  // Единичные векторы радиального, восточного и северного направлений
  e_1 = r / Norm(r);
  e_2 = Cross ( Vec3D(0, 0, 1), r );
  e_2 = e_2 / Norm(e_2);
  e_3 = Cross (e_1, e_2);
  // Позиционный угол
  c = Dot(d, e_3);
  s = Dot(d, e_2);
  phi = atan2(s, c);
  return Modulo(phi, 2.0*pi);
}
```

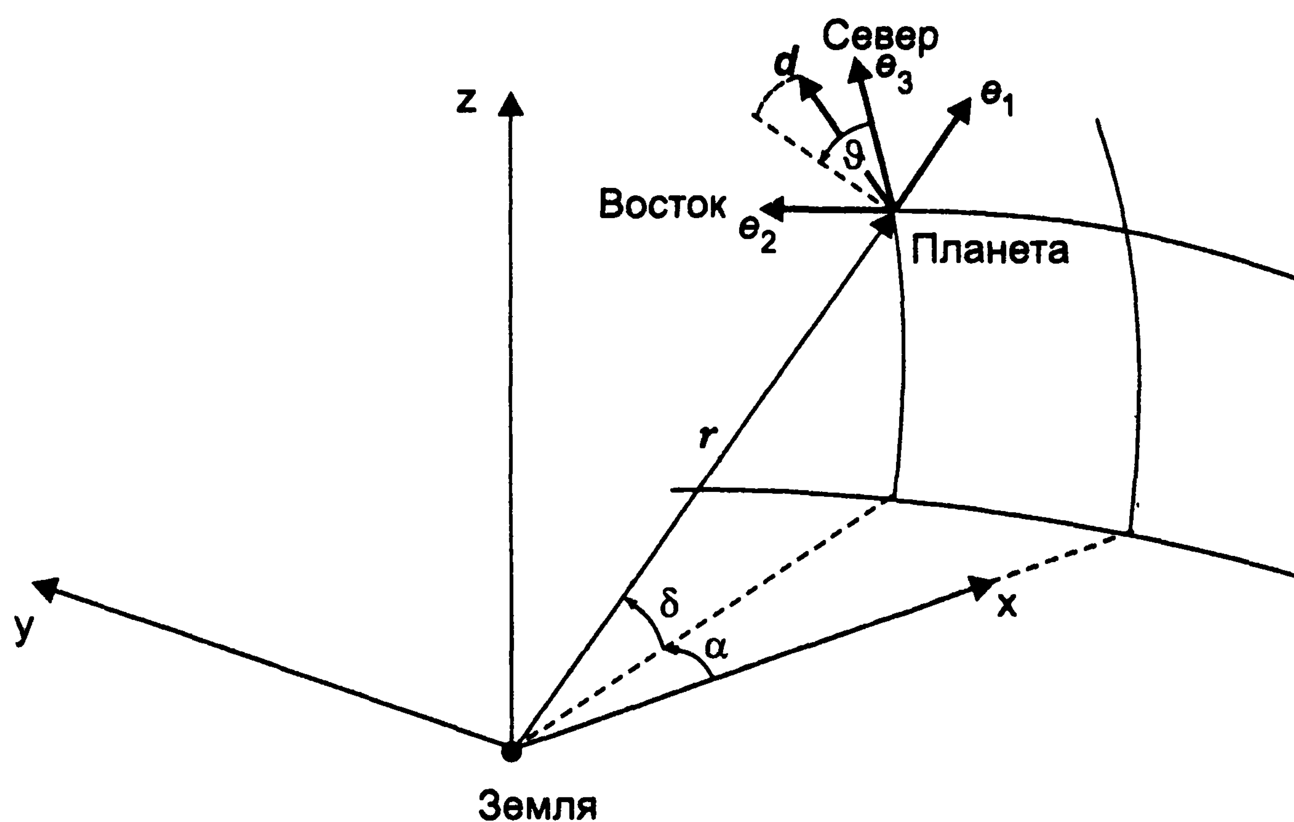


Рис. 7.2. Вычисление позиционного угла

Если в качестве вектора d подставить планетоцентрические координаты Солнца, то функцию PosAng можно использовать и для определения позиционного угла Солнца. Мы подробнее обсудим это в разделе 7.2 в связи с вопросом об определении фаз планет.

7.1.2. Планетографические координаты

Подобно тому, как на Земле построена система географических координат, мы можем построить систему сферических координат, привязанную к поверхности любой другой планеты. Этой системой координат можно пользоваться для задания положения видимых на поверхности деталей. Опорной плоскостью системы координат служит экватор небесного тела. Поскольку большинство планет Солнечной системы сплюснуты вследствие своего вращения, необходимо различать два определения широты:

- Плането*центрическая* широта φ' — это угол между направлением на заданную точку и плоскостью планетного экватора. Он соответствует склонению в геоцентрической экваториальной системе координат.

- Плането*графическая* широта φ отличается от планетоцентрической φ′ в тех случаях, когда планета сплюснута. Она соответствует углу между линией отвеса, перпендикулярной поверхности планеты в выбранной точке, и плоскостью ее экватора. В случае Земли она соответствует географической широте (рис. 9.5).

Связь между геоцентрической широтой φ′ и географической широтой φ описывается в разделе 9.3. В нашем случае нужно просто подставить экваториальный радиус планеты R_{Eq} вместо радиуса Земли $R_{\oplus}$ и вычислить значение сплюснутости планетного эллипсоида:

$$f = \frac{R_{Eq} - R_{Pol}}{R_{Eq}} . \quad (7.5)$$

Связь между планетоцентрическими φ′ и планетографическими φ широтами задается уравнением

$$\operatorname{tg} \varphi' = (1 - f)^2 \operatorname{tg} \varphi . \quad (7.6)$$

Данные об экваториальных радиусах и сплюснутости планет, необходимые для вычисления физических эфемерид, мы поместили в функцию Shape, поскольку эти данные совершенно независимы от вычислительной процедуры.

```
//------------------------------------------------------------------------
// Shape: Выдает значения экваториального радиуса и сплюснутости планеты
//   Planet    Идентификатор планеты (см. APC_Planets.h)
//   R_equ     Экваториальный радиус в [км]
//   f         Геометрическая сплюснутость
//------------------------------------------------------------------------
void Shape (PlanetType Planet, double& R_equ, double& f)
{
  switch (Planet) {
    case Sun:      R_equ =696000.0;  f = 0.0;        break;
    case Mercury:  R_equ =   2439 0; f = 0.0;        break;
    case Venus:    R_equ =   6051.0; f = 0.0;        break;
    case Earth:    R_equ =   6378.14; f = 0 00335281; break;
    case Mars:     R_equ =   3393.4; f = 0.0051865;  break;
    case Jupiter:  R_equ = 71398.0,  f = 0.0648088,  break;
    case Saturn:   R_equ = 60000.0,  f = 0.1076209;  break;
    case Uranus:   R_equ = 25400.0;  f = 0 030;      break,
    case Neptune:  R_equ = 24300.0,  f = 0 0259;     break,
    case Pluto:    R_equ =   1500 0; f = 0.0;
  }
}
```

Здесь, как и прежде, для идентификации планет используется перечисление

```
enum PlanetType { Sun, Mercury, Venus, Earth, Mars,
                  Jupiter, Saturn, Uranus, Neptune, Pluto },
```

Как и в случае широты, существует различие между планетоцентрической и планетографической долготами.

- Плането*центрическая* долгота λ′ увеличивается в направлении к востоку независимо от направления вращения планеты.

- Плането*графическая* долгота λ отсчитывается в направлении, противоположном вращению планеты. Из этого определения следует, что долгота центрального меридиана, проходящего через центр видимого с Земли диска планеты, с течением времени всегда увеличивается.

Для Венеры, Урана и Плутона, которые имеют обратное вращение, оба определения долготы идентичны ($\lambda = \lambda'$), тогда как для других планет, которые подобно Земле вращаются с запада на восток, планетоцентрические и планетографические долготы различаются знаком ($\lambda = -\lambda'$). Особо надо отметить, что в случае Солнца гелиографическая долгота увеличивается в направлении вращения и поэтому совпадает с гелиоцентрической долготой, несмотря на то, что Солнце имеет прямое вращение.

Для установления точки отсчета долгот (начального меридиана) по возможности стараются использовать хорошо заметные детали на поверхности планеты, но если это невозможно, то прибегают к иным способам. Так, для планет-гигантов — Юпитера, Сатурна, Урана и Нептуна, — вращение которых не является твердотельным, вводятся системы координат вращающиеся с постоянной угловой скоростью. Например, в районе экватора атмосфера Юпитера вращается с периодом 9^h30^m, а в более высоких широтах с периодом 9^h56^m. Соответственно для экваториальных районов ($|\varphi| \le 9°$) используется система координат I, а для всех остальных— система II. Радиоастрономические исследования внешних планет показывают, что периоды вращения их магнитных полей не совпадают с теми, что определяются оптическими методами. Причина состоит в том, что магнитное поле генерируется в глубинных слоях планет, которые, по всей видимости, вращаются твердотельно и совершенно независимо от движения атмосферных слоев. Таким образом, для описания вращения относительно планетного радиоизлучения необходимо ввести систему III. Для Урана и Нептуна в настоящее время используется только система III. Для Сатурна в дополнение к системе III также используется система I, но ее роль в основном состоит в том, чтобы обеспечивать привязку для старых рисунков и фотографий.

Для описания положения начального меридиана планеты используется угол W. Он отсчитывается:

- от точки пересечения плоскости земного экватора (J2000) с плоскостью экватора планеты (точка Q)
- вдоль экватора планеты
- в направлении ее вращения
- до точки пересечения экватора планеты с начальным меридианом (точка B).

Из двух точек, в которых пересекаются экваторы Земли и планет, точка Q выбирается так, чтобы проходящие через нее точки поверхности планеты двигались, пересекая плоскость земного экватора с юга на север (рис. 7.3). Угол W увеличивается пропорционально времени, и его можно вычислить, опираясь на данные табл. 7.2. Параметр

$$d = \mathrm{JD} - 2451545 \qquad (7.7)$$

обозначает число дней, прошедших от стандартной эпохи J2000.

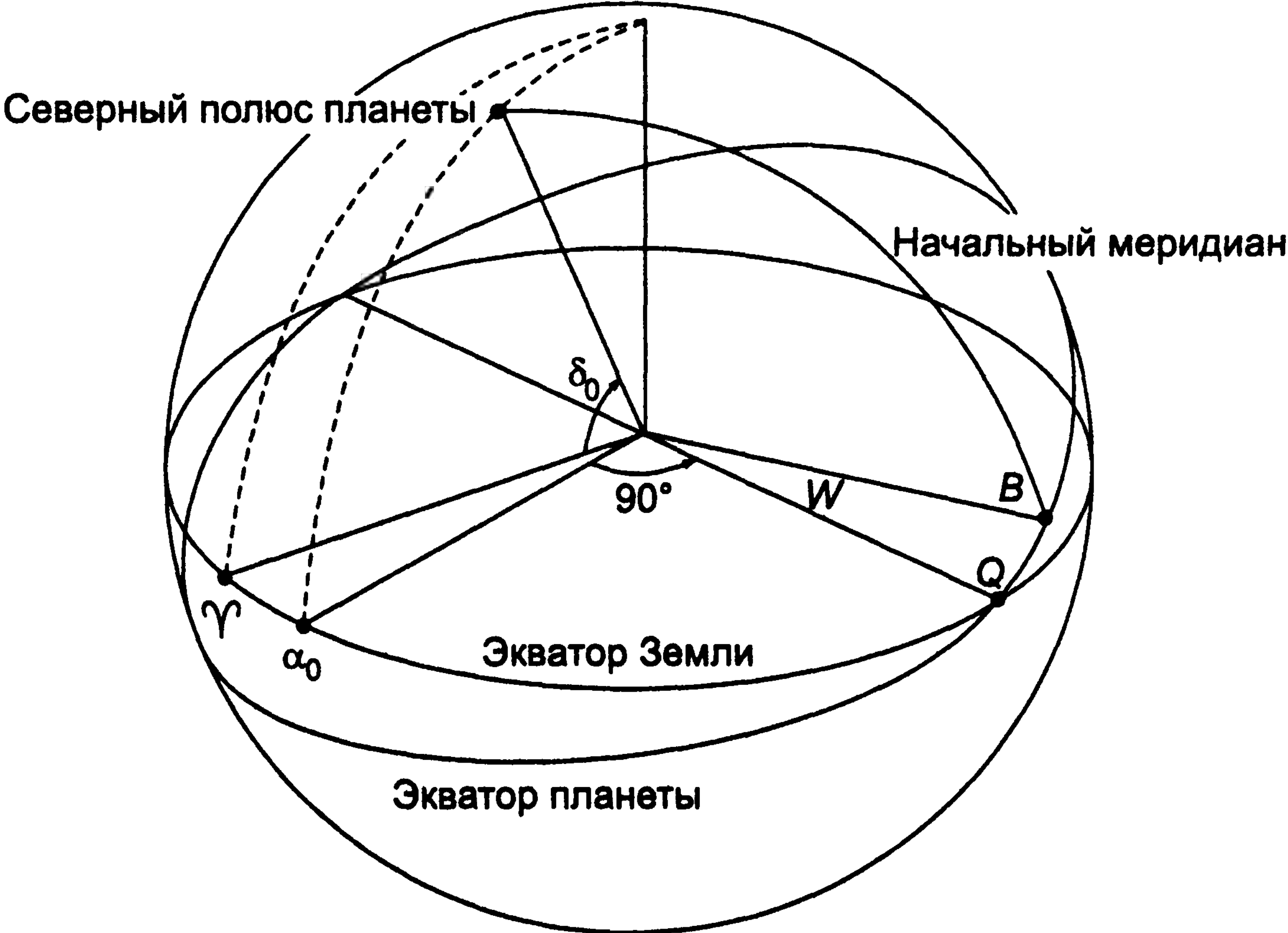


Рис. 7.3. Ориентация начального меридиана

Таблица 7.2. Ориентация начального меридиана Солнца и планет (МАС, 1994)

		W (J2000)
Солнце		84°182 + 14°1844000*d*
Меркурий		329°68 + 6°1385025*d*
Венера		160°20 − 1°4813688*d*
Земля		190°16 + 360°9856235*d*
Марс		176°901 + 350°8919830*d*
Юпитер,	система I	67°1 + 877°900*d*
	система II	43°3 + 870°270*d*
	система III	284°695 + 870°536*d*
Сатурн,	система I	227°2037+ 844°300*d*
	система III	38°90 + 810°7939024*d*
Уран,	система III	203°81 − 501°1600928*d*
Нептун,	система III	253°18 + 536°3128492*d* − 0°48 sin(*N*)
Плутон		236°77 − 56°3623195*d*

Функция `Orient` определяет численные значения параметров α_0, δ_0 и W, используемых для описания положе́ния оси вращения и начального меридиана планеты. Она устроена так, что позволяет легко учесть любые возможные будущие изменения принятых значений параметров вращения. Наряду с временем и идентификатором планеты функция `Orient` требует указания номера используемой вращающейся системы координат. Для этих целей используется параметр `System`, значения которого берутся из перечисления

```
enum SystemType  { Sys_I. Sys_II. Sys_III };
```

Для Юпитера и Сатурна в зависимости от значения этого параметра функция `Orient` вычисляет положение начального меридиана для экваториальных районов, высоких широт или для радиоизлучения. Обратите внимание на дополнительный параметр `Sense`, возвращаемый функцией `Orient`, который принимает значения типа

```
enum RotationType { Direct, Retrograde };
```

Он указывает, является вращение планеты прямым или обратным.

```
//-------------------------------------------------------------------------------
// Orient: Определяет ориентацию планетоцентрической системы координат
//         и направление вращения
//   Planet    Идентифицирует планету (см. APC_Planets.h)
//   System    Указывает тип системы координат (только для Юпитера и Сатурна)
//   T         Время в юлианских столетиях от эпохи J2000
//   E         Матрица преобразования от земных экватора и равноденствия
//             эпохи J2000 к системе координат, связанной с экватором
//             и начальным меридианом планеты
//   Sense     Направление вращения (прямое или обратное)
//-------------------------------------------------------------------------------
void Orient ( PlanetType Planet, SystemType System, double T,
              Mat3D& E, RotationType& Sense )
{
  double N, RA, Dec, W;
  double d = 36525.0*T;
  // Прямое восхождение и склонение оси вращения
  // относительно экватора и равноденствия J2000;
  // Ориентация начального меридиана
  switch (Planet) {
    case Sun:      RA  = 286.13;
                   Dec =  63.87;
                   W   =  84.182 +  14.1844000*d; break;
    case Mercury:  RA  = 281.01  -   0 033*T;
                   Dec =  61.45  -   0.005*T;
                   W   = 329.68  +   6.1385025*d; break;
    case Venus.    RA  = 272.76;
                   Dec =  67.16;
                   W   = 160.20  -   1.4813688*d; break;
    case Earth:    RA  =   0.00  -   0.641*T;
                   Dec =  90.00  -   0.557*T;
                   W   = 190.16  + 360.9856235*d; break;
    case Mars:     RA  = 317.681 -   0.108*T;
                   Dec =  52.886 -   0 061*T;
                   W   = 176.901 + 350.8919830*d; break;
    case Jupiter.  RA  = 268.05  -   0.009*T;
                   Dec =  64.49  +   0.003*T;
                   switch (System) {
                     case Sys_I   : W =  67 10  + 877 900*d, break;
                     case Sys_II  : W =  43.30  + 870.270*d; break;
                     case Sys_III : W = 284.695 + 870.536*d, break;
                   }
                   break;
    case Saturn:   RA  =  40.589 -   0.036*T,
                   Dec =  83.537 -   0.004*T;
                   switch (System) {
                     case Sys_I
```

```
                case Sys_II  : W = 227.2037 + 844.3000000*d. break.
                case Sys_III : W =  38 90   + 810 7939024*d. break.
              }
              break;
  case Uranus:  RA  = 257.311.
                Dec = -15.175;
                W   = 203.81   - 501.1600928*d: break. // System III
  case Neptune: N   = Rad*(357 85+52 316*T).
                RA  = 299.36   + 0 70*sin(N).
                Dec =  43.46   - 0.51*cos(N).
                W   = 253.18   + 536 3128492*d - 0 48*sin(N). break.
  case Pluto    RA  = 313 02;
                Dec =   9.09;
                W   = 236 77   -  56.3623195*d.
 }
 RA*=Rad. Dec*=Rad; W=Rad*Modulo(W.360.0);

 // Преобразование от средних земных экватора и равноденствия J2000
 // к системе координат. связанной с фиксированным экватором и
 // начальным меридианом планеты
 E = R_z(W) * R_x(pi/2.0-Dec) * R_z(pi/2.0+RA);

 // Направление вращения
 if ( Planet==Venus || Planet==Uranus || Planet==Pluto )
   Sense = Retrograde.
 else
   Sense = Direct.
}
```

После того как мы определили положение оси вращения и начального меридиана планеты, мы можем вычислить планетоцентрические координаты Солнца и Земли, которые важно знать при наблюдении планет. Эти координаты определяют, например, как наклонен северный полюс планеты по отношению к Земле, какая точка планеты находится в центре видимого диска планеты, в какой точке планеты Солнце находится в зените и его лучи падают вертикально.

Для вычисления планетоцентрической долготы Земли или Солнца мы будем использовать систему координат, вращающуюся вместе с планетой, задаваемую тремя взаимно перпендикулярными единичными векторами $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$. Вектор $\boldsymbol{e}_3$ расположим параллельно оси вращения, а вектор $\boldsymbol{e}_1$ направим из центра планеты к точке пересечения ее экватора и начального меридиана. Выпишем покомпонентно экваториальные координаты этих векторов

$$\boldsymbol{e}_1 = \begin{pmatrix} -\cos W \sin\alpha_0 - \sin W \sin\delta_0 \cos\alpha_0 \\ +\cos W \cos\alpha_0 - \sin W \sin\delta_0 \sin\alpha_0 \\ +\sin W \cos\delta_0 \end{pmatrix},$$

$$\boldsymbol{e}_2 = \begin{pmatrix} +\sin W \sin\alpha_0 - \cos W \sin\delta_0 \cos\alpha_0 \\ -\sin W \cos\alpha_0 - \cos W \sin\delta_0 \sin\alpha_0 \\ +\cos W \cos\delta_0 \end{pmatrix}, \tag{7.8}$$

$$\boldsymbol{e}_3 = \begin{pmatrix} +\cos\delta_0 \cos\alpha_0 \\ +\cos\delta_0 \sin\alpha_0 \\ +\sin\delta_0 \end{pmatrix}.$$

Данные выражения легко получить, зная соотношения между векторами Гаусса $\boldsymbol{P}$, $\boldsymbol{Q}$ и $\boldsymbol{R}$ (см. разд. 4.5), если заменить элементы орбиты i, Ω и ω соответственно: углом между земным и планетным экваторами $90° - \delta_0$, прямым восхождением линии пересечения плоскостей земного и планетного экваторов $\alpha_0 + 90°$ и уголом между этой линией и начальным меридианом W. В продолжение этой аналогии матрицу

$$\boldsymbol{E} = \begin{pmatrix} \boldsymbol{e}_1^T \\ \boldsymbol{e}_2^T \\ \boldsymbol{e}_3^T \end{pmatrix} = \boldsymbol{R}_z(W)\boldsymbol{R}_x(\pi/2 - \delta)\boldsymbol{R}_z(\pi/2 + \alpha) \tag{7.9}$$

для преобразования экваториальных координат в систему координат, связанную с планетой, можно получить произведением трех матриц элементарных поворотов.

Пусть $\boldsymbol{r} = (x, y, z)$ — экваториальные координаты Земли относительно центра планеты. Тогда проектируя $\boldsymbol{r}$ на направления векторов $\boldsymbol{e}_1$, $\boldsymbol{e}_2$ и $\boldsymbol{e}_3$, мы получим декартовы планетоцентрические координаты

$$\begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} = \begin{pmatrix} -\boldsymbol{e}_1 \cdot \boldsymbol{r} \\ -\boldsymbol{e}_2 \cdot \boldsymbol{r} \\ -\boldsymbol{e}_3 \cdot \boldsymbol{r} \end{pmatrix} = \begin{pmatrix} r\cos\varphi'\cos\lambda' \\ r\cos\varphi'\sin\lambda' \\ r\sin\varphi' \end{pmatrix}. \tag{7.10}$$

Отсюда, учитывая, что

$$\begin{aligned} \operatorname{tg}\varphi' &= \frac{s_z}{\sqrt{s_x^2 + s_y^2}}, \\ \operatorname{tg}\lambda' &= \frac{s_y}{s_x}, \\ \operatorname{tg}\varphi &= \frac{\tan\varphi'}{(1-f)^2}, \\ \lambda &= \begin{cases} +\lambda' & \text{(вращение обратное)}, \\ -\lambda' & \text{(вращение прямое)}, \end{cases} \end{aligned} \tag{7.11}$$

можно получить значения планетоцентрических и планетографических долгот и широт Земли.

Планетоцентрические и планетографические координаты Солнца можно получить в точности таким же способом. Достаточно заменить планетоцентрические координаты Земли соответствующими координатами Солнца.

Все перечисленные шаги, необходимые для вычисления планетографических координат, реализуются функцией `Rotation`. Кроме планетографических долготы и широты, которые обычно приводятся в ежегодниках, эта функция возвращает также планетоцентрическую широту, которая понадобится нам в дальнейшем для вычисления освещенности. Если на вход вместо координат Земли подать планетоцентрические координаты Солнца, то мы получим планетографические координаты Солнца.

```
//---------------------------------------------------------------------------
// Rotation: Вычисляет параметры вращения планеты
//   r        Планетоцентрические координаты Земли, отнесенные к земным
//            экватору и равноденствию J2000
//   E        Матрица преобразования от средних экватора и равноденствия Земли (J2000)
//            к системе координат, связанной с экватором и начальным меридианом планеты
//   Sense    Направление вращения (прямое или обратное)
//   f        Геометрическая сплюснутость
//   lon      Планетографическая долгота Земли [рад]
//   lat_g    Планетографическая широта Земли [рад]
//   lat_c    Планетоцентрическая широта Земли [рад]
//---------------------------------------------------------------------------
void Rotation ( const Vec3D& r, const Mat3D& E, RotationType Sense, double f,
                double& lon, double& lat_g, double& lat_c )
{
  Vec3D s;
  // Планетоцентрические широта и долгота
  s = E*r;  lat_c = s[theta];  lon  = s[phi];
  // Планетографические широта и долгота
  if (Sense==Direct) lon=-lon;
  lon = Modulo(lon,2*pi);
  lat_g = atan ( tan(lat_c) / ((1.0-f)*(1.0-f)) );
}
```

7.2. Условия освещения

Помимо данных об ориентации планеты наблюдателю важно знать условия ее освещения. Они описываются фазой, позиционным углом Солнца и блеском планеты.

7.2.1. Фаза и элонгация

Поскольку планеты Солнечной системы сами по себе не испускают видимого излучения, а светят отраженным солнечным светом, обычно светлой бывает только часть видимого с Земли диска планеты. Мерой освещенной части служит угол между планетоцентрическими векторами, задающими направления на Солнце и на Землю. Этот угол называется *фазовым углом i*. Значение фазового угла $i = 0°$ соответствует полностью освещенному диску планеты, а значение $i = 90°$ — диску, освещенному наполовину.

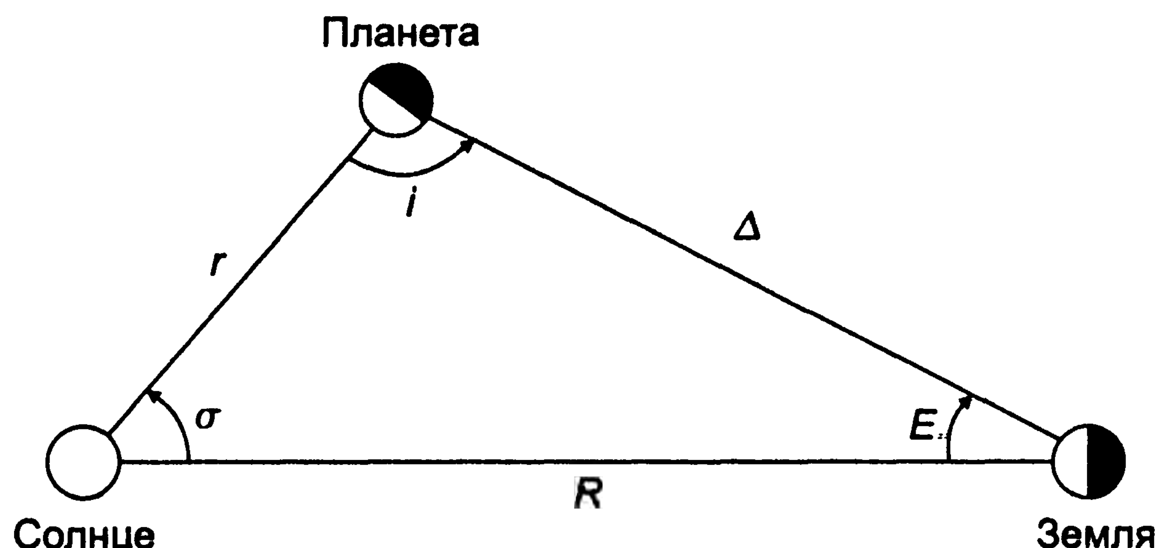


Рис. 7.4. Фазовый угол i и элонгация E

В треугольнике Солнце–Земля–планета (рис. 7.4) обозначим расстояние от Солнца до планеты r, расстояние от Солнца до Земли — R, а от планеты до Земли — Δ. Тогда фазовый угол можно определить по теореме косинусов

$$\cos i = \frac{\Delta^2 + r^2 - R^2}{2r\Delta}. \tag{7.12}$$

Освещенная доля планетного диска называется фазой. Ее значение можно получить по формуле

$$k = \frac{1 + \cos i}{2}. \tag{7.13}$$

Обратите внимание, что k относится не к сферической поверхности планеты, а к ее видимому диску.

Так же как мы нашли по теореме косинусов фазовый угол, можно получить значение *элонгации* E

$$\cos E = \frac{\Delta^2 + R^2 - r^2}{2R\Delta}, \tag{7.14}$$

которая характеризует угловое расстояние между Солнцем и планетой при наблюдении с Земли. Знание элонгации позволяет грубо оценить условия наблюдения планеты — не слишком ли близко к Солнцу расположена она на небе.

Уравнения, необходимые для вычисления элонгации и фазы планеты, объединены в функции `Illum`. На входе она получает гелиоцентрические координаты планеты и Земли, которые должны быть представлены в одной системе координат и приведены к одной эпохе.

```
//------------------------------------------------------------------------
// Illum· Вычисляет параметры освещения планеты
//   r          Гелиоцентрическое положение планеты
//   r_Earth    Гелиоцентрическое положение Земли
//   Elong      Элонгация в [рад]
//   phi        Фазовый угол в [рад]
//   k          Фаза
// Примечание· r и r_Earth должны относиться к одной системе координат и эпохе
//------------------------------------------------------------------------
void Illum ( const Vec3D& r, const Vec3D& r_Earth,
             double& Elong, double& phi, double& k )
{
  Vec3D   r_geoc;
  double  R,RE,D,c_phi;
  r_geoc = r - r_Earth;   // Геоцентрическое положение планеты
  R  = Norm(r);           // Расстояние Солнце-планета
  RE = Norm(r_Earth);     // Расстояние Солнце-Земля
  D  = Norm(r_geoc),      // Расстояние Земля-планета
  // Элонгация, фазовый угол, фаза
  Elong = acos ( (D*D + RE*RE - R*R) / (2 0*D*RE) );
  c_phi = (D*D + R*R - RE*RE) / (2.0*D*R);
  phi   = acos (c_phi);
  k     = 0 5*(1 0+c_phi);
}
```

7.2.2. Позиционный угол Солнца

Поскольку планеты двигаются не точно в плоскости эклиптики, солнечный свет редко падает на их поверхности точно с востока или запада (в земной системе координат). В общем случае разделительная линия между освещенной и темной частями диска планеты (терминатор) не проходит через полюса видимого диска.

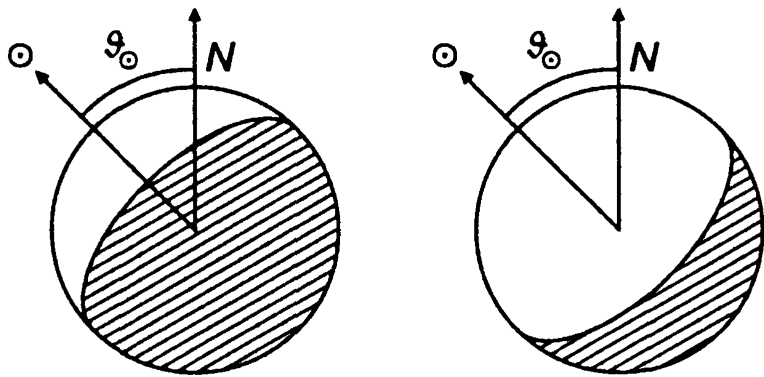


Рис. 7.5. Позиционный угол Солнца при различных фазах

Ориентация освещенной части диска по отношению к северу наблюдателя однозначно описывается позиционным углом $\vartheta_\odot$ Солнца, который иногда называют углом освещения (рис. 7.5). Вычислить позиционный угол Солнца можно, пользуясь полученным ранее соотношением (7.4) для позиционного угла оси вращения, если подставить в него вместо направляющего вектора оси вектор, направленный от планеты к Солнцу.

Таблица 7.3. Видимые звездные величины планет

Планета	$V(1,0)$	$\Delta m(i)$
Меркурий	$-0^{m}.42$	$+3^{m}.80\cdot(i/100°)-2^{m}.73\cdot(i/100°)^2+2^{m}.00\cdot(i/100°)^3$
Венера	$-4^{m}.40$	$+0^{m}.09\cdot(i/100°)+2^{m}.39\cdot(i/100°)^2-0^{m}.65\cdot(i/100°)^3$
Земля	$-3^{m}.86$	
Марс	$-1^{m}.52$	$+1^{m}.60\cdot(i/100°)$
Юпитер	$-9^{m}.40$	$+0^{m}.50\cdot(i/100°)$
Сатурн	$-8^{m}.88$	$-2^{m}.60\cdot\lvert\sin\varphi'_\odot\rvert+1^{m}.25\cdot\lvert\sin\varphi'_\odot\rvert^2+4^{m}.40\cdot\lvert\lambda'_\oplus-\lambda'_\odot\rvert$
Уран	$-7^{m}.19$	
Нептун	$-6^{m}.87$	
Плутон	$-1^{m}.0$	

Это позволяет использовать для вычисления угла освещения уже знакомую нам функцию PosAng. Достаточно подставить геоцентрические экваториальные координаты планеты в качестве вектора $\boldsymbol{r}$ и планетоцентрические экваториальные координаты Солнца вместо направляющего вектора $\boldsymbol{d}$. Последние проще всего получить, поменяв знак гелиоцентрических координат планеты, которые все равно вычисляются по ходу дела. Поскольку позиционный угол обычно отсчитывается от направления на север в момент наблюдения, координаты Солнца и планеты следует приводить к равноденствию текущей даты.

7.2.3. Видимая звездная величина

Под видимой звездной величиной планеты мы понимаем общий блеск видимой с Земли освещенной части диска планеты, выраженный в звездных величинах. Он определяется расстоянием планеты от Солнца и от Земли, а также размером и составом поверхности или атмосферы планеты. Абсолютной звездной величиной планеты называют блеск планеты при фиксированных эталонных условиях освещения ($V(1{,}0)$), то есть на расстоянии по 1 а. е. от Солнца и от Земли при фазовом угле $i=0°$. Ее значения варьируются от $-0\overset{m}{.}4$ для Меркурия до $-9\overset{m}{.}4$ для Юпитера.

Освещенность поверхности планеты обратно пропорциональна квадрату ее расстояния от Солнца. Аналогично освещенность, создаваемая на Земле планетой, находящейся на фиксированном расстоянии от Солнца, уменьшается обратно пропорционально квадрату расстояния до Земли. Уменьшение освещенности в 100 раз соответствует увеличению звездной величины на 5^m. С учетом этого видимую звездную величину планеты можно определить по формуле

$$m = V(1{,}0) + 5\log\left(\frac{r\cdot\Delta}{\mathrm{AU}^2}\right) + \Delta m(i). \qquad (7.15)$$

Здесь член $\Delta m(i)$ описывает зависимость звездной величины от фазы. Он учитывает как уменьшение освещенной части диска, то есть фазы, так и зависимость рассеяния света от угла освещения. В общем случае $\Delta m(i)$ описывается эмпирически определяемым степенным законом.

В случае Сатурна необходимо дополнительно принимать во внимание вклад колец в общий блеск системы. Этот вклад можно представить как функцию планетоцентрической широты Солнца $\varphi'_\odot$ и разности планетоцентрических долгот Солнца и Земли $\lambda'_\odot - \lambda'_\oplus$.

Значения абсолютной звездной величины планет $V(1{,}0)$ и зависимость блеска от фазы $\Delta m(i)$ представлены в табл. 7.3. Функция `Bright` использует эти данные для определения видимой звездной величины планет.

```
//------------------------------------------------------------------------------
// Bright: Вычисляет видимую звездную величину планеты
//   Planet     Идентифицирует планету (см. APC_Planets.h)
//   r          Гелиоцентрическое расстояние планеты [а.е.]
//   Delta      Расстояние от планеты до Земли [а.е.]
//   phi        Фазовый угол [рад]
//   lat        Планетоцентрическая широта Солнца [рад]
//   Dlong      Разность планетоцентрических долгот Солнца и Земли [рад]
// <return>:    Блеск планеты в звездных величинах [m]
// Примечание: lat и Dlong нужно задавать только для Сатурна.
//              В остальных случаях они игнорируются
//------------------------------------------------------------------------------
double Bright ( PlanetType Planet, double r, double Delta, double phi,
                double lat, double Dlong )
{
  // Переменные
  double  p, sd, dl, mag;
  // Нормализованный фазовый угол
```

```
  p = Deg*phi/100.0;
  // Блеск на единичном расстоянии
  switch ( Planet ) {
    case Sun:      return 0.0;
    case Mercury:  mag = -0.42+(3.80-(2.73-2.00*p)*p)*p; break;
    case Venus:    mag = -4.40+(0.09+(2.39-0.65*p)*p)*p; break;
    case Earth:    mag = -3.86;        break;
    case Mars:     mag = -1.52+1.6*p; break;
    case Jupiter:  mag = -9.40+0.5*p; break;
    case Saturn:
      sd = fabs(sin(lat));
      dl = fabs(Modulo(Deg*Dlong+180.0,360.0)-180.0)/100.0;
      mag = -8.88 - 2.60*sd + 1.25*sd*sd + 4.40*dl;
      break;
    case Uranus:   mag = -7.19; break;
    case Neptune:  mag = -6.87; break;
    case Pluto:    mag = -1.0;
  }
  // Видимый блеск
  return ( mag + 5.0*log10(r*Delta) );
}
```

7.2.4. Диаметр видимого диска

Диаметр видимого диска планеты — это угол, под которым планета видна с Земли. Его можно определить, зная экваториальный радиус R_{Eq} планеты и расстояние Δ до нее:

$$\varnothing_{Eq} = 2 \cdot \arcsin\left(\frac{R_{Eq}}{\Delta}\right). \qquad (7.16)$$

Для планет, имеющих заметную сплюснутость, следует различать экваториальный и полярный диаметры. Видимый полярный диаметр всегда меньше экваториального и зависит от величины сплюснутости и от планетоцентрической широты $\varphi'_{\oplus}$ Земли. При малых значениях сплюснутости видимый полярный диаметр можно в первом приближении вычислить по формуле

$$\varnothing_{Pol} = (1 - f\cos^2\varphi'_{\oplus})\varnothing_{Eq}. \qquad (7.17)$$

7.3. Программа Phys

Программа Phys вычисляет таблицу физических эфемерид больших планет и Солнца. Она использует описанные выше функции для определения планетографических координат Земли, условий освещения и позиционного угла северного конца оси вращения планеты.

Необходимые координаты тел Солнечной системы, приведенные к равноденствию даты, вычисляются с помощью функции PertPosition по разложениям в аналитические ряды. Эффекты нутации и аберрации малы и ими можно пренебречь при решении данных задач с точностью, достаточной для практических це-

лей. А вот поправку за время движения света от планеты к Земле учитывать необходимо. Поэтому сначала вычисляется геометрическое расстояние междуЗемлей и планетой, по которому определяется световая задержка, а затем координаты и ориентация планеты на момент, когда наблюдаемый на Земле свет был испущен небесным телом.

Для получения планетографических и гелиографических координат Земли нет необходимости вносить поправку за прецессию для приведения к равноденствию даты. Все используемые значения определяются только взаимным расположением небесных тел и потому не зависят от используемой системы координат. Только при вычислении позиционного угла по отношению к направлению на северный полюс мира нам необходимо выполнить соответствующее преобразование координат.

После вычисления физических эфемерид планет определяются видимый диаметр Солнца, позиционный угол его оси и гелиографические координаты Земли. Ориентация планетных систем координат относительно земной системы экваториальных координат определяется функцией Orient и напрямую передается функции Rotation. Геоцентрические координаты Солнца, необходимые также для вычисления позиционного угла, определяются путем обращения знака гелиоцентрических координат Земли.

Остается добавить, что гелиографическая долгота отсчитывается в противоположном направлении по сравнению с планетографическими долготами. Гелиографическая долгота Земли всегда увеличивается. Кроме того, позиционный угол солнечной оси принято приводить к диапазону от –180° до 180°. Для этого в программу добавлены соответствующие инструкции.

```
//------------------------------------------------------------------------------
// File:    Phys cpp
// Purpose. Физические эфемериды планет и Солнца
// (c) 1999 Oliver Montenbruck. Thomas Pfleger
//------------------------------------------------------------------------------

#include <cmath>
#include <iomanip>
#include <iostream>
#include "APC_Const.h"
#include "APC_Math h"
#include "APC_Phys h"
#include "APC_Planets.h"
#include "APC_PrecNut h"
#include "APC_Spheric.h"
#include "APC_Sun h"
#include "APC_Time h"
#include "APC_VecMat3D.h"
using namespace std;

//------------------------------------------------------------------------------
// Основная программа
//------------------------------------------------------------------------------
void main()
{
  // Названия объектов
```

```
const char* Name[] =
  { "Sun", "Mercury", "Venus", "Earth", "Mars",
    "Jupiter", "Saturn", "Uranus", "Neptune", "Pluto" };

// Переменные
PlanetType    Planet;
int           iPlanet, year, month, day;
double        hour, T, T0;
Vec3D         r, r_Earth, r_geoc;
double        Delta,  R_equ, D_equ, f,
Mat3D         E_I, E_II, E_III,  P;
Vec3D         d,
RotationType  Sense;
double        long_I, long_II, long_III, lat_g, lat_c;
double        long_Sun, latg_Sun, latc_Sun;
double        Elong, phi, k, mag,  PosAng_Ax, PosAng_Sun;

// Заголовок
cout << endl
     << " PHYS: physical ephemerides of the planets and the Sun" << endl
     << "       (c) 1999 Oliver Montenbruck, Thomas Pfleger    " << endl
     << endl;
// Приглашение для ввода данных
cout << " Date (yyyy mm dd hh.hhh) ... ";
cin >> year >> month >> day >> hour; cin.ignore(81,'\n');
T = (Mjd(year,month,day)+hour/24.0-MJD_J2000)/36525.0;
// Шапка таблицы
cout << endl
     << "            D      V      i    PA(S)   PA(A)"
     << "    L(I)  L(II) L(III)     B" << endl
     << "           '  \"     mag     o      o       o  "
     << "     o      o      o       o" << endl;

// Прецессия (от текущей даты до EME2000)
P = PrecMatrix_Equ(T,T_J2000);
// Гелиоцентрическое положение Земли
// относительно среднего экватора и равноденствия даты
r_Earth = Ecl2EquMatrix(T) * PertPosition(Earth,T);
// Цикл по объектам
for (iPlanet=Sun; iPlanet<=Pluto; iPlanet++) {
  Planet = (PlanetType) iPlanet;
  if (Planet==Earth) continue;  // Skip Earth
  // Геометрическое геоцентрическое положение и световая задержка
  r = Ecl2EquMatrix(T) * PertPosition(Planet,T);
  Delta = Norm(r-r_Earth),
  T0 = T - (Delta/c_light)/36525 0,
  // Исправленное за световую задержку положение планеты
  r = Ecl2EquMatrix(T) * PertPosition(Planet,T0);
  // Геоцентрическое положение
  r_geoc = r - r_Earth;
  // Радиус видимого диска ["]
  Shape ( Planet, R_equ,f );
  D_equ = Arcs*2.0*asin(R_equ/(Delta*AU));
  // Преобразование от средних экватора и равноденствия даты
  // к системе координат, связанной с небесным телом
  Orient ( Planet, Sys_I  ,T0, E_I  , Sense ),  E_I   = E_I   * P,
  Orient ( Planet, Sys_II ,T0, E_II , Sense ),  E_II  = E_II  * P,
```

```
Orient ( Planet, Sys_III.T0, E_III, Sense );  E_III = E_III * P;
// Планетоцентрические долгота и широта Земли
Rotation ( -r_geoc, E_I,   Sense, f, long_I,  lat_g,lat_c );
Rotation ( -r_geoc, E_II,  Sense, f, long_II, lat_g,lat_c );
Rotation ( -r_geoc, E_III, Sense, f, long_III,lat_g,lat_c );
// Позиционный угол оси вращения
PosAng_Ax  = PosAng ( r_geoc, Row(E_I,z) );
// Специальные вычисления для Солнца и планет
if (Planet==Sun) {
  // Преобразования долготы и позиционного угла оси вращения
  // с учетом особенностей их задания для Солнца
  long_I = 2.0*pi-long_I;
  if (PosAng_Ax>pi) PosAng_Ax-=2.0*pi;
}
else  {
  // Освещенность и видимый блеск
  Illum ( r, r_Earth, Elong,phi,k );
  if (Planet==Saturn) {
    // Планетоцентрические долгота и широта Солнца
    Rotation ( -r, E_I, Sense, f, long_Sun, latg_Sun, latc_Sun );
    mag = Bright ( Planet, Norm(r), Norm(r_geoc), phi,
                   latc_Sun, long_Sun-long_I );
  }
  else
    mag = Bright ( Planet, Norm(r),Norm(r_geoc), phi);
  // Позиционный угол Солнца
  PosAng_Sun = PosAng ( r_geoc, -r );
};

// Вывод
if (Planet==Sun)
  cout << " " << left << setw(6) << Name[Planet] << right << fixed
       << setprecision(2) << setw(8) << Angle(D_equ/60.0,DMMm)
       << setprecision(2) << setw(29) << Deg*PosAng_Ax;
else
  cout << " " << left << setw(9) << Name[Planet] << right << fixed
       << setprecision(2) << setw(5) << D_equ
       << setprecision(1) << setw(6) << mag
       << setprecision(1) << setw(7) << Deg*phi
       << setprecision(2) << setw(8) << Deg*PosAng_Sun
       << setprecision(2) << setw(8) << Deg*PosAng_Ax;
// Вывод планетографической долготы Земли
switch (Planet) {
  // Тела, для которых определена только система I
  case Sun: case Mercury: case Venus: case Mars: case Pluto:
    cout << setprecision(2) << setw(8) << Deg*long_I
         << setw(14) << " ";
    break;
  case Jupiter:
    cout << setprecision(2) << setw(8) << Deg*long_I
         << setprecision(2) << setw(7) << Deg*long_II
         << setprecision(2) << setw(7) << Deg*long_III;
    break;
  case Saturn:
    cout << setprecision(2) << setw(8)  << Deg*long_I
         << setprecision(2) << setw(14) << Deg*long_III;
    break;
```

```
      // Тела, для которых определена только система III
      case Uranus· case Neptune:
        cout << setprecision(2) << setw(22) << Deg*long_III:
    }
    cout << setprecision(2) << setw(8)  << Deg*lat_g << endl:
  } // Конец цикла по планетам
}
```

В качестве примера вычислим физические эфемериды на 0^h эфемеридного времени 30 августа 1999 года. Как обычно, данные, вводимые пользователем, выделены курсивом.

Программа выводит следующие данные: видимые экваториальный диаметр D в угловых секундах, видимую звездную величину V, фазовый угол i, а также позиционный угол Солнца PA(S) и северного полюса PA(A). Дополнительно вычисляются планетографические координаты Земли в системах координат, используемых для каждой планеты.

```
PHYS: physical ephemerides of the planets and the Sun
      (c) 1999 Oliver Montenbruck, Thomas Pfleger

Date (yyyy mm dd hh hhh) ... 1999 08 30 00 0

                D      V      i    PA(S)   PA(A)    L(I)  L(II) L(III)      B
              ' "    mag      o      o       o       o      o      o        o
Sun      31 40.64                          20.43  203.33                   7.15
Mercury    5.30   -1.4   31.6  119.79    21.10  328.45                   5.31
Venus     55.62   -4.2  156.5   79.44    17.13  334.31                   8.68
Mars       7.95    0.3   44.2  283.48    37.85   29.70                  12.81
Jupiter   45.22   -2.7   10.0   71.42   339.34  329 80 176.12  24.39     3.78
Saturn    18.73    0.2    5.9   74.63   359.41   93 75         118 66  -25 77
Uranus     3.69    5.7    1.1  251.46   262.90                 155.83  -34.84
Neptune    2.29    7.9    1.1  257.45   355.90                 256.25  -28.71
Pluto      0.14   13.8    1.9  279.11    75.60  311 55                 -22.30
```

Венера находится вблизи нижнего соединения в нескольких градусах к югу от Солнца и выглядит как крупный, но очень тонкий серп ($i = 156^\circ\!.5$). В этой фазе при благоприятных условиях наблюдения Венеру можно увидеть даже днем. Солнечный свет падает на Венеру почти точно с востока ($\vartheta_0 = 79^\circ\!.4$).

Юпитер обращает на себя внимание большим видимым диаметром и блеском. В это время он близок к поворотной точке, после которой начинается попятное движение, и постепенно приближается к противостоянию, которое случится через два месяца.

А вот противостояние Марса уже осталось в прошлом, и его фаза ($i = 44^\circ\!.2$) довольно заметно отличается от полной. Малый видимый диаметр — немного меньше восьми угловых секунд — делает детали поверхности практически неразличимыми.

8. Орбита Луны

8.1. Общие свойства лунной орбиты

Земля и Солнце оказывают основное влияние на движение Луны. Легко убедиться в том, что наибольшее гравитационное воздействие на Луну оказывает не ее ближайший сосед — Земля, а более далекое Солнце. Сила тяготения убывает пропорционально квадрату расстояния, тем не менее притяжение огромной массы Солнца превосходит притяжение Земли. Расстояние между Землей и Луной $\mathrm{r} \approx 380\,000$ км, Солнце находится на расстоянии $\mathrm{R} \approx 150$ млн км, отношение масс Солнца и Земли $\mathrm{M/m} \approx 330\,000$. Можно вычислить, во сколько раз притяжение Солнца превосходит земное:

$$\frac{F_{\odot}}{F_{\oplus}} = \frac{M}{R^2} \Big/ \frac{m}{r^2} \approx 2.$$

Благодаря взаимному расположению Солнца, Земли и Луны сумма обеих сил всегда имеет составляющую, направленную к Солнцу. Движение Луны поэтому может быть представлено как обращение на расстоянии около 150 млн км вокруг Солнца по эллиптической орбите, на которую накладываются незначительные месячные колебания. Поскольку сила притяжения никогда не бывает направлена от Солнца, несмотря на эти колебания, орбита Луны всегда обращена вогнутостью к Солнцу.

Хотя лунная орбита определяется в первую очередь притяжением Солнца, не кажется целесообразным описывать ее в гелиоцентрических координатах. Нас прежде всего интересует движение Луны относительно Земли. Земля и Луна как ближайшие соседи почти в равной мере испытывают гравитационное влияние Солнца. Если бы это влияние было в точности одинаковым для обеих планет, оно бы никак не сказывалось на их взаимном расположении. В этом случае притяжение Солнца определяло бы лишь их общее годичное движение, а притяжение Земли определяло бы месячное движение Луны. Разница сил, действующих со стороны Солнца на Луну в различных ее положениях относительно Земли, очень мала — в двести раз меньше силы притяжения Луны Землей. Таким образом, удобнее всего описывать движение Луны относительно центра Земли. В этой системе координат Луна обращается вокруг Земли почти по кеплеровской орбите, с месячным периодом. Отклонения от кеплеровской орбиты вызваны влиянием Солнца.

Лунная орбита имеет эксцентриситет $e = 0{,}055$ и наклонение к эклиптике около $5{,}^\circ1$. Средний орбитальный период — от перигея до перигея, известный под названием *аномалистического* месяца, составляет 27,55 дней. Это почти на двое

суток меньше *синодического* месяца — периода, соответствующего времени между двумя одинаковыми фазами, например последовательными новолуниями или полнолуниями, и равного 29,53 дням. Разница возникает в результате видимого годичного движения Солнца относительно звезд, составляющего около 30° в месяц. Совершившей полный оборот вокруг Земли Луне требуется еще около двух суток, чтобы «догнать» Солнце и предстать перед земным наблюдателем в той же фазе.

Эксцентричность орбиты вызывает отклонения Луны от среднего положения до 6°,3. Эта наиболее заметная неравномерность ее движения известна под названием *уравнения центра* и не имеет ничего общего с возмущениями от Солнца. Величина этого возмущения следует из уравнения центра (6.1), рассмотренного в связи с разложениями в ряд выражений для планетных орбит. Птолемей в *Альмагесте* отмечает, что учета этого явления недостаточно для удовлетворительного описания лунной орбиты. Он первым обнаружил явление *эвекции* — отклонения, достигающего 1°,3, которое зависит от взаимного положения Луны и Солнца. Значительно позже, но еще до того, как Ньютон заложил фундамент теоретического объяснения движения Луны с помощью закона всемирного тяготения, Тихо Браге, производивший гораздо более точные наблюдения, обнаружил два других явления — *вариацию* и *годичное неравенство*. В 1770 году Майер впервые опубликовал лунные таблицы, имевшие достаточную точность для определения координат и времени в открытом море. В настоящее время известно более тысячи периодических возмущений лунной орбиты.

Влияние Солнца вызывает не только короткопериодические колебания лунной орбиты, но и целый ряд вековых явлений, среди которых наиболее важно медленное вращение плоскости орбиты Луны. Линия узлов, то есть линия пересечения лунной орбиты с плоскостью эклиптики, совершает полный оборот за 18,6 лет, однако наклонение остается практически неизменным. Вращение линии узлов является *обратным*, то есть долгота восходящего узла *убывает* на 20°[1] в год относительно точки весеннего равноденствия. Причина такого смещения заключена в том, что притяжение Солнца стремится привести Луну в плоскость эклиптики и установить перпендикулярно ей ось симметрии лунной орбиты. Подобно оси гироскопа, на который действуют внешние силы, эта ось совершает медленное вращение вокруг полюса эклиптики с указанным периодом в 18,6 лет.

Положение перигея или, что то же самое, ориентация большой оси в пространстве также меняется со временем. Смещение перигея по орбите — прямое, превышающее 40°[2] в год, таким образом, примерно за 8,5 лет он совершает полный оборот.

Ниже приводятся общепринятые обозначения *основных аргументов* теории движения Луны. При

$$T = \frac{\text{JD} - 2451545}{36525}$$

[1] 19°,3. — *Примеч. перев.*
[2] 40°,7. — *Примеч. перев.*

- средняя долгота Луны (L_0)

$$L_0 = 218^\circ\!.31617 + 481267^\circ\!.88088 \cdot T - 4''\!.06 \cdot T^2 = \qquad (8.1)$$
$$= 0^r\!.60643382 + 1336^r\!.85522467 \cdot T - 0^r\!.00000313 \cdot T^2;$$

- средняя аномалия Луны (l)

$$l = 134^\circ\!.96292 + 477198^\circ\!.86753^\circ \cdot T + 33''\!.25 \cdot T^2 = \qquad (8.2)$$
$$= 0^r\!.37489701 + 1325^r\!.55240982 \cdot T + 0^r\!.00002565 \cdot T^2;$$

- средняя аномалия Солнца[1] (l')

$$l' = 357^\circ\!.52543 + 35999^\circ\!.04944^\circ \cdot T - 0''\!.58 \cdot T^2 = \qquad (8.3)$$
$$= 0^r\!.99312619 + 99^r\!.99735956 \cdot T - 0^r\!.00000044 \cdot T^2;$$

- средняя долгота Луны, отсчитываемая от восходящего узла орбиты F

$$F = 93^\circ\!.27283 + 483202^\circ\!.01873^\circ \cdot T - 11''\!.56 \cdot T^2 = \qquad (8.4)$$
$$= 0^r\!.25909118 + 1342^r\!.22782980 \cdot T - 0^r\!.00000892 \cdot T^2;$$

- средняя элонгация Луны D — разница средних долгот Солнца и Луны

$$D = 297^\circ\!.85027 + 445267^\circ\!.11135 \cdot T - 5''\!.15 \cdot T^2 = \qquad (8.5)$$
$$= 0^r\!.82736186 + 1236^r\!.85308708 \cdot T - 0^r\!.00000397 \cdot T^2.$$

Значения приведены в градусах и единицах оборотов ($1^r = 360^\circ$). В формуле для L_0, в отличие от остальных углов, учтено влияние прецессии. Долгота восходящего узла ☊ не определена, она находится из разницы ☊ $= L_0 - F$.

Истинная эклиптическая долгота Луны λ отличается от средней долготы на величину $\Delta\lambda$, являющуюся функцией от l, l', F и D:

$$\lambda = L_0 + \Delta\lambda,$$

где

$\Delta\lambda =$	$+22640'' \cdot \sin(l)$	(уравнение центра)
	$-4586'' \cdot \sin(l - 2D)$	(эвекция)
	$+2370'' \cdot \sin(2D)$	(вариация)
	$+769'' \cdot \sin(2l)$	(уравнение центра)
	$-668'' \cdot \sin(l')$	(годичное неравенство)
	$-412'' \cdot \sin(2F)$	(редукция к эклиптике)
	$-212'' \cdot \sin(2l - 2D)$	
	$-206'' \cdot \sin(l + l' - 2D)$	
	$+192'' \cdot \sin(l + 2D)$	

[1] В русскоязычной литературе этот параметр называется средней аномалией Земли. — *Примеч. перев.*

$-165'' \cdot \sin(l' - 2D)$

$+148'' \cdot \sin(l - l')$

$-125'' \cdot \sin(D)$ (параллактическое неравенство)

$-110'' \cdot \sin(l + l')$

$-55'' \cdot \sin(2F - 2D)$

В случае невозмущенной орбиты эклиптическая широта β определяется наклонением орбиты i и положением Луны на орбите:

$$\sin\beta = \sin i \cdot \sin u,$$

$$\beta \approx i \cdot \sin u.$$

Здесь u — аргумент широты, а именно угол между позиционным вектором и линией узлов. С учетом возмущений со стороны Солнца широта Луны может быть представлена в первом приближении как

$$\beta \approx 18520'' \sin(S) + N,$$

где $S = F + \Delta S$. ΔS определяется по формуле, большинство входящих в нее членов те же самые, что и для $\Delta\lambda$. С учетом важнейших отличий получим:

$$\Delta S \approx \Delta\lambda + 412'' \sin(2F) + 541'' \sin(l').$$

Величина N представляет собой поправку на небольшой ряд отклонений по широте, вызванных колебаниями наклонения орбиты:

$$N = -526'' \sin(F - 2D) + 44'' \sin(l + F - 2D) - 31'' \sin(-l + F - 2D) \ldots \quad (8.6)$$

Изложенное позволяет составить общее представление о направлении развития теории движения Луны и о том, каким же образом вычисляются ее эклиптические широта и долгота. Приведенные выше уравнения использованы в программе MiniMoon, упомянутой в главе 3. Эта программа поможет уяснить последовательность действий при вычислениях.

Отметим, что до сих пор мы сознательно не приводили разложение в ряд для непосредственного вычисления эклиптической широты β. Если учитывать наиболее важные члены, этот ряд выглядит следующим образом:

$$\beta = 18461'' \sin(F) + 1010'' \sin(l + F) + 1000'' \sin(l - F) - 624'' \sin(F - 2D) -$$
$$- 199'' \sin(l - F - 2D) - 167'' \sin(l + F - 2D) + \ldots \, .$$

Несмотря на ее простоту, использование этой формулы связано с неоправданно громоздкими вычислениями. Это объясняется тем, что каждый ее член содержит F в качестве множителя в нечетном количестве, тогда как члены в разложении для $\Delta\lambda$ и ΔS всегда включают F только в четном количестве. Каждому члену в $\Delta\lambda$ соответствует член в ΔS, содержащий ту же самую тригонометрическую функцию. Вычисление ΔS при правильно составленной программе позволяет обой-

тись без лишних тригонометрических расчетов. Напротив, ряды для $\Delta\lambda$ и $\Delta\beta$ не содержат членов, которые могли бы вычисляться совместно.

8.2. Теория Брауна

В настоящее время наиболее популярна теория, созданная Е. В. Брауном[1] в начале XX века. Это аналитическое представление движения Луны. Мы приводим здесь ее изложение в Nautical Almanac Office's *Improved Lunar Ephemeris* (1954). В программе MoonPos учитывается лишь часть из более чем тысячи рассматриваемых в теории возмущений. Координаты вычисляются с точностью до одной секунды дуги.

Вычисления начинаются с определения средних аргументов L, l, l', F и D по формулам (8.1–8.5). Время T выражается в юлианских столетиях от эпохи J2000. Средние аргументы также испытывают влияние небольших долгопериодических колебаний, для учета которых необходимо вносить следующие поправки:

ΔL_0	Δl	$\Delta l'$	ΔF	ΔD		
+0″,84	+2″,94	+6″,40	+0″,21	+7″,24	$\cdot\sin(2\pi(0{,}19833 + 0{,}05611T))$	
+0″,31	+0″,31	0″,00	+0″,31	+0″,31	$\cdot\sin(2\pi(0{,}27869 + 0{,}04508T))$	
+14″,27	+14″,27	0″,00	+14″,27	+14″,27	$\cdot\sin(2\pi(0{,}16827 - 0{,}36903T))$	(8.7)
+7″,26	+9″,34	0″,00	−88″,70	+7″,26	$\cdot\sin(2\pi(0{,}34734 - 5{,}37261T))$	
+0″,28	+1″,12	0″,00	−15″,30	+0″,28	$\cdot\sin(2\pi(0{,}10498 - 5{,}37899T))$	
+0″,24	+0″,83	+1″,89	+0″,24	+2″,13	$\cdot\sin(2\pi(0{,}42681 - 0{,}41855T))$	
0″,00	0″,00	0″,00	−1″,86	0″,00	$\cdot\sin(2\pi(0{,}14943 - 5{,}37511T))$	

Используя уточненные значения средних аргументов, вычисляют пять рядов возмущений:

$\Delta\lambda$ — возмущения в эклиптической долготе

ΔS, N, $\gamma_1 C$ — возмущения в эклиптической широте

$\Delta\sin\Pi$ — возмущения в параллаксе

Все эти ряды имеют общую структуру:

$$\begin{Bmatrix}\Delta\lambda, \Delta S, N\\ \gamma_1 C, \Delta\sin\Pi\end{Bmatrix} = \sum_n \begin{Bmatrix} a_n, b_n, c_n\\ d_n, e_n\end{Bmatrix}\cdot\begin{Bmatrix}\sin\\ \cos\end{Bmatrix}(p_n l + q_n l' + r_n F + s_n D).$$

Массив (p, q, r, s) определяет зависимость индивидуальных слагаемых от l, l', F и D и, следовательно, периодичность выражения. Удобнее производить вычисле-

[1] Несколько ранее, а точнее в 1895 г. — *Примеч. перев.*

ния, комбинируя все члены, имеющие общие характеристики. Далее приводится небольшая часть таблицы всех членов, описывающих возмущения.

$\Delta\lambda$	ΔS	$\gamma_1 C$	$\Delta\sin\Pi$	p	q	r	s
+22639,"500	+22609,"07	+0,"079	+186,"5398	+1	0	0	0
−4586,"465	−4578,"13	−0,"077	+34,"3117	+1	0	0	−2
+2369,"912	+2373,"36	+0,"601	+28,"2333	0	0	0	+2
+769,"016	+767,"96	+0,"107	+10,"1657	+2	0	0	0
−668,"146	−126,"98	−1,"302	−0,"3997	0	+1	0	0
−411,"608	−0,"20	+0,"000	−0,"0124	0	0	+2	0

Из-за различия коэффициентов ряды для величины N не могут вычисляться совместно с остальными рядами. Впрочем, это выражение для N содержит всего лишь нескольких членов.

Число членов, описывающих возмущения, велико, поэтому лучше отдельно вычислить встречающиеся в выражениях тригонометрические функции. Вычисление синуса или косинуса, в сравнении такими простейшими операциями, как четыре арифметических действия, требует относительно большого времени. Однако нет необходимости вычислять каждую функцию в отдельности, если использовать тригонометрические формулы:

$$\cos(\alpha_1 + \alpha_2) = \cos\alpha_1 \cos\alpha_2 - \sin\alpha_1 \sin\alpha_2,$$

$$\sin(\alpha_1 + \alpha_2) = \sin\alpha_1 \cos\alpha_2 + \cos\alpha_1 \sin\alpha_2.$$

С помощью краткой функции AddThe

```
void AddThe ( double c1, double s1, double c2, double s2,
              double& c, double& s )
{
  c = c1 * c2 - s1 * s2;
  s = s1 * c2 + c1 * s2;
}
```

мы можем сперва вычислить значения синусов и косинусов, на которые умножаются средние аргументы.

$$\text{Cos[j+o][i]} = \begin{cases} \cos(jl)\,(i=0) \\ \cos(jl')\,(i=1) \\ \cos(jF)\,(i=2) \\ \cos(jD)\,(i=3) \end{cases}, \qquad \text{Sin[j+o][i]} = \begin{cases} \sin(jl)\,(i=0) \\ \sin(jl')\,(i=1) \\ \sin(jF)\,(i=2) \\ \sin(jD)\,(i=3) \end{cases}.$$

При l $(i = 0)$ и используя

```
const int o   = 6;            // индекс смещения

Cos[o][i]=1.0;  Cos[o+1][i]=cos(arg)*fac;  Cos[o-1][i]=+Cos[o+1][i];
    Sin[o][i]=0.0;  Sin[o+1][i]=sin(arg)*fac;  Sin[o-1][i]=-Sin[o+1][i];

    for (int j=2;j<=max;j++) {
      AddThe ( Cos[o+j-1][i],Sin[o+j-1][i], Cos[o+1][i],Sin[o+1][i],
               Cos[o+j][i],Sin[o+j][i] );
```

```
    Cos[o-j][i]=+Cos[o+j][i];
    Sin[o-j][i]=-Sin[o+j][i];
  };
```

лишь дважды вычислить синус или косинус, чтобы определить все величины $\cos(jl)$ и $\sin(jl)$ для промежутка $j = -\max... + \max$. При реализации рядов ILE в C++ рекомендуется определить специальный класс ILE_Pert, содержащий среди прочего члены Cos и Sin.

```
// Constants - Постоянные
const int o   = 6;          // Индекс смещения
const int dim = 2*o+1;      // Размерность рабочего массива

// Определение класса ILE_Pert для суммирования
// Брауновских возмущений лунной орбиты.
//
class ILE_Pert
{

  public:

    //Инициализация (средние аргументы, долгопериодические поправки и т. д.)
    void Init (double T);

    // Член, описывающий возмущение
    void Term (int p, int q, int r, int s, double& x, double& y);

    // Суммирование возмущений от Солнца
    void AddSol ( double coeffl, double coeffS, double coeffg,
                  double coeffP, int p, int q, int r, int s );

    // Суммирование возмущений в широте
    void AddN (double coeffN, int p, int q, int r, int s);

    // Планетные возмущения
    void Planetary (double T);

    // Координаты
    double lambda();  // эклиптическая долгота, рад
    double beta();    // эклиптическая широта, рад
    double dist();    // геоцентрическое расстояние, км

  private:

    double Dgam;                        // Долгопериодическое возмущение
    double Dlam, DS, gamlC, sinPi, N;   // Периодическое возмущение
    double L0, l,ls,F,D;                // Средние аргументы теории движения Луны
    double Cos[dim][4], Sin[dim][4];    // Косинусы и синусы средних аргументов
};
```

Метод Term, использующий уже вычисленные значения Cos [] [] и Sin [] [], позволяет легко получить

$$x = \cos(pl + ql' + rF + sD) \text{ и } y = \sin(pl + ql' + rF + sD)$$

для заданных коэффициентов (p, q, r, s).

```
// Term· вычисление x=cos(p*l+q*ls+r*F+s*D) и y=sin(p*l+q*ls+r*F+s*D)

  void ILE_Pert::Term (int p, int q, int r, int s, double& x, double& y)
  {
    int i[4];

    i[0]=p; i[1]=q; i[2]=r; i[3]=s;  x=1.0; y=0.0;

    for (int k=0; k<=3; k++)
      if (i[k]!=0) AddThe(x,y,Cos[o+i[k]][k],Sin[o+i[k]][k],x,y);
  }
```

Действие этой функции легко пояснить на примере вычисления рядов для возмущений в широте N. Метод AddN разработан для суммирования всех членов ряда, в который разложено выражение для N.

```
// AddN: Суммирование возмущений в широте
  //
  void ILE_Pert::AddN (double coeffN, int p, int q, int r, int s)
  {
    double x,y;


    Term(p,q,r,s,x,y);
    N += coeffN*y;
  }
```

Вначале Term вычисляет соответствующие тригонометрические функции (в случае N это только синусы). Затем полученные значения умножаются на соответствующие коэффициенты и складываются. Отдельный член ряда вычисляется при каждом обращении к методу AddN, который воспринимает соответствующий коэффициент и величины (p, q, r, s) как входные параметры. Естественно, перед обращением к AddN все значения массивов Sin [] [] и Cos [] [] должны быть определены, что выполняется методом Init, вычисляющим вспомогательные данные для заданного времени T. Фрагмент кода для вычислений возмущений в широте N выглядит так:

```
ILE_Pert Pert;
  Pert.Init(T);
  // Возмущения от Солнца в широте
  Pert.AddN(-526.069, 0, 0,1,-2);   Pert.AddN(  -3.352, 0, 0,1,-4);
  Pert.AddN( +44.297,+1, 0,1,-2);   Pert.AddN(  -6.000,+1, 0,1,-4);
  Pert.AddN( +20.599,-1, 0,1, 0);   Pert.AddN( -30.598,-1, 0,1,-2);
  Pert.AddN( -24.649,-2, 0,1, 0);   Pert.AddN(  -2.000,-2, 0,1,-2);
  Pert.AddN( -22.571, 0,+1,1,-2);   Pert.AddN( +10.985, 0,-1,1,-2);
```

Метод AddSol вычисляет ряды $\Delta\lambda$, ΔS, $\gamma_1 C$ и $\Delta \sin\Pi$ точно таким же способом.

```
// AddSol: Суммирование возмущений от Солнца
  //
  void ILE_Pert::AddSol (
    double coeffl, double coeffS, double coeffg, double coeffP,
    int p, int q, int r, int s )
  {
    double x,y;
    Term (p,q,r,s,x,y);
```

```
    Dlam  += coeffl*y: DS    += coeffS*y:
    gamlC += coeffg*x: sinPi += coeffP*x;
  }
```

Описанный метод не является исчерпывающим, поскольку, строго говоря, коэффициенты при членах рядов несколько отличаются от приведенных выше значений. Каждый из коэффициентов при $\Delta\lambda$, ΔS, N, $\gamma_1 C$ и $\Delta\sin\Pi$ в действительности является функцией различных параметров как, например, эксцентриситет солнечной орбиты. Браун присваивает этим функциям определенное численное значение. Чтобы учесть эти параметры, каждый член (характеризующийся массивом (p, q, r, s)) следует умножить на коэффициент, получаемый по формуле

$$(1{,}000002208)^{|p|}\cdot(1{,}0-0{,}002495388(T+1))^{|q|}\cdot(1{,}000002707+139{,}978\Delta\gamma)^{|r|}.$$

Величина $\Delta\gamma$ вычисляется с помощью ILE_Pert::Init по следующей формуле

$$\begin{aligned}\Delta\gamma = &-0{,}000003332\cdot\sin(2\pi(0{,}59734-5{,}37261T))-\\ &-0{,}000000539\cdot\sin(2\pi(0{,}35498-5{,}37899T))-\\ &-0{,}000000064\cdot\sin(2\pi(0{,}39943-5{,}37511T)).\end{aligned}$$

Все эти поправки можно легко учесть, если при вычислении массивов Cos [] [] Sin [] [] полученные величины умножить на соответствующие коэффициенты:

Cos [o+1] [0] и Sin [o+1] [0] на (1,000002208),

Cos [o+1] [1] и Sin [o+1] [1] на (1,0 – 0,002495388(T+1)),

Cos [o+1] [2] и Sin [o+1] [2] на (1,000002708 + 139,978 $\Delta\gamma$).

Соответствующие поправочные коэффициенты учитываются при вычислении всех тригонометрических функций при обращении к AddThe. Например, Sin [o+3,1] присваивается значение $(1{,}0-0{,}002495388(T+1))^{|3|}\cdot\sin(3l')$, а не $\sin(3l')$. Это позволяет автоматически и без особого труда получить правильные значения членов разложения.

Таким образом вычисляются различные возмущения, вызванные притяжением Солнца. Однако Солнце не единственное тело в Солнечной системе, влияющее на лунную орбиту. Полная теория движения Луны не может пренебрегать влиянием планет. Функция MoonPos учитывает наиболее значительные возмущения, вызываемые Венерой и Юпитером:

$$\begin{aligned}\Delta\lambda_{plan} = &+0{,}82''\sin(0{,}7736^r-62{,}5512^rT)+0{,}31''\sin(0{,}0466^r-125{,}1025^rT)+\\ &+0{,}35''\sin(0{,}5785^r-25{,}1042^rT)+0{,}66''\sin(0{,}4591^r+1335{,}8075^rT)+\\ &+0{,}64''\sin(0{,}3130^r-91{,}5680^rT)+1{,}14''\sin(0{,}1480^r+1331{,}2898^rT)+\\ &+0{,}21''\sin(0{,}5918^r+1056{,}5859^rT)+0{,}44''\sin(0{,}5784^r+1322{,}8595^rT)+\\ &+0{,}24''\sin(0{,}2275^r-5{,}7374^rT)+0{,}28''\sin(0{,}2965^r+2{,}6929^rT)+\\ &+0{,}33''\sin(0{,}3132^r+6{,}3368^rT).\end{aligned}$$

Эти возмущения в долготе вычисляются ILE_Pert::Planetary.

Теперь все необходимые величины известны. Чтобы получить значение эклиптической долготы Луны, следует к значению средней долготы прибавить значения солнечных и планетных возмущений:

$$\lambda = L_0 + \Delta\lambda + \Delta\lambda_{план}.$$

L_0 здесь – значение средней долготы, исправленное в соответствии с формулой (8.7). При вычислении эклиптической широты β вначале значение ΔS прибавляется к значению среднего расстояния от узла F:

$$S = F + \Delta S.$$

Тогда мы имеем

$$\beta = (1{,}000002708 + 139{,}978\Delta\gamma)\cdot\{18519''\!,70 + \gamma_1 C\}\cdot\sin(S) - \{6''\!,24\}\cdot\sin(3S) + N.$$

Расстояние Луны получается из значения синуса горизонтального параллакса:

$$r = (1/\sin\Pi)R_{\oplus} \quad (\text{ радиус Земли } R_{\oplus} \approx 6378{,}14 \text{ км}).$$

sin П имеет значение

$$\sin\Pi = 0{,}999953253\cdot(3422''\!,7 + \Delta\sin\Pi)\cdot\frac{\pi}{180\cdot 3600''}.$$

Множитель

$$\frac{\pi}{180\cdot 3600''} = \frac{1}{206264''\!,81}$$

служит для перевода секунд дуги в радианы.

Ниже приводится полный вариант функции MoonPos, включая не воспроизведенные ранее методы ILE_Pert. Для заданного времени T, выраженного в юлианских столетиях, прошедших от эпохи JD2000, MoonPos вычисляет геоцентрические эклиптические координаты Луны (в километрах) соответственно среднему равноденствию даты.

```
// Синус
//
double Sine (double x) { return sin(pi2*Frac(x)); }

// Инициализация
//
void ILE_Pert::Init (double T)
{
  //
  // Переменные
  //
  double dL0, dl, dls, dF, dD;          // Долгопериодические возмущения
  double T2, arg, fac;                  // Вспомогательные переменные
  double S1, S2, S3, S4, S5, S6, S7;
  int    max;

  T2=T*T; // Время
```

```
  // «Сброс» возмущений
  Dlam=0.0; DS=0.0; gamlC=0.0; sinPi=3422.7000; N=0.0;

  // Долгопериодические возмущения
  S1 = Sine (0.19833+0.05611*T);  S2 = Sine (0.27869+0.04508*T);
  S3 = Sine (0.16827-0.36903*T);  S4 = Sine (0.34734-5.37261*T);
  S5 = Sine (0.10498-5.37899*T);  S6 = Sine (0.42681-0.41855*T);
  S7 = Sine (0.14943-5.37511*T);

  dL0 = 0.84*S1+0.31*S2+14.27*S3+ 7.26*S4+ 0.28*S5+0.24*S6;
  dl  = 2.94*S1+0.31*S2+14.27*S3+ 9.34*S4+ 1.12*S5+0.83*S6;
  dls =-6.40*S1                                   -1.89*S6;
  dF  = 0.21*S1+0.31*S2+14.27*S3-88.70*S4-15.30*S5+0.24*S6-1.86*S7;
  dD  = dL0-dls;

  Dgam   = -3332e-9 * Sine (0.59734-5.37261*T)
            -539e-9 * Sine (0.35498-5.37899*T)
             -64e-9 * Sine (0.39943-5.37511*T);

  // Средние аргументы теории движения Луны (включая долгопериодические поправки)
  // L0 - средняя долгота Луны
  // l - средняя аномалия Луны     l' mean anomaly of the Sun - средняя аномалия Солнца
  // F - среднее расстояние от узла
  // D - средняя элонгация от Солнца

  L0 = pi2*Frac(0.60643382+1336.85522467*T-0.00000313*T2) + dL0/Arcs;
  l  = pi2*Frac(0.37489701+1325.55240982*T+0.00002565*T2) + dl /Arcs;
  ls = pi2*Frac(0.99312619+  99.99735956*T-0.00000044*T2) + dls/Arcs;
  F  = pi2*Frac(0.25909118+1342.22782980*T-0.00000892*T2) + dF /Arcs;
  D  = pi2*Frac(0.82736186+1236.85308708*T-0 00000397*T2) + dD /Arcs;

  // Косинусы и синусы, на которые умножаются значения средних аргументов;
  // включая вековую поправку
  for (int i=0; i<=3; i++) {
    switch(i) {
      case 0: arg=l;  max=4; fac=1.000002208;               break;
      case 1: arg=ls; max=3; fac=0.997504612-0.002495388*T; break;
      case 2: arg=F;  max=4; fac=1.000002708+139.978*Dgam;  break;
      case 3: arg=D;  max=6; fac=1.0;                       break;
    };

    Cos[o][i]=1.0;  Cos[o+1][i]=cos(arg)*fac;  Cos[o-1][i]=+Cos[o+1][i];
    Sin[o][i]=0.0;  Sin[o+1][i]=sin(arg)*fac;  Sin[o-1][i]=-Sin[o+1][i];

    for (int j=2;j<=max;j++) {
      AddThe ( Cos[o+j-1][i],Sin[o+j-1][i], Cos[o+1][i],Sin[o+1][i],
               Cos[o+j][i],Sin[o+j][i] );
      Cos[o-j][i]=+Cos[o+j][i];
      Sin[o-j][i]=-Sin[o+j][i];
    };
  };
}

//
// Term: вычисление x=cos(p*l+q*ls+r*F+s*D) и y=sin(p*l+q*ls+r*F+s*D)
//
//
```

```
void ILE_Pert::Term (int p, int q, int r, int s, double& x, double& y)
{
  int i[4];

  i[0]=p; i[1]=q; i[2]=r; i[3]=s;  x=1.0; y=0.0;
  for (int k=0; k<=3; k++)
    if (i[k]!=0) AddThe(x,y,Cos[o+i[k]][k],Sin[o+i[k]][k],x,y);

}

//
// AddSol: суммирование возмущений от Солнца
//
void ILE_Pert::AddSol (
  double coeffl, double coeffS, double coeffg, double coeffP,
  int p, int q, int r, int s )
{
  //
  // Переменные
  //
  double x,y;

  Term (p,q,r,s,x,y);
  Dlam  += coeffl*y; DS    += coeffS*y;
  gam1C += coeffg*x; sinPi += coeffP*x;
}

//
// AddN: суммирование возмущений в широте
//
void ILE_Pert::AddN (double coeffN, int p, int q, int r, int s)
{
  //
  // Переменные
  //
  double x,y;

  Term(p,q,r,s,x,y);
  N += coeffN*y;
}

//
// Возмущения в эклиптической широте от Венеры и Юпитера
//
//
void ILE_Pert::Planetary (double T)
{
  Dlam +=
        +0.82*Sine(0.7736  -62.5512*T)+0.31*Sine(0.0466 -125.1025*T)
        +0.35*Sine(0.5785  -25.1042*T)+0.66*Sine(0.4591+1335.8075*T)
        +0.64*Sine(0.3130  -91.5680*T)+1.14*Sine(0.1480+1331.2898*T)
        +0.21*Sine(0.5918+1056.5859*T)+0.44*Sine(0.5784+1322.8595*T)
        +0.24*Sine(0.2275   -5.7374*T)+0.28*Sine(0.2965   +2.6929*T)
        +0.33*Sine(0.3132   +6.3368*T);
}

//
```

```
  // lambda, beta, dist
  //
  double ILE_Pert::lambda()
  {
    return Modulo ( L0+Dlam/Arcs, pi2 );
  }

  double ILE_Pert::beta()
  {
    //
    // Переменные
    //
    double S   = F + DS/Arcs;
    double fac = 1.000002708+139.978*Dgam;

    return (fac*(18518.511+1.189+gam1C)*sin(S)-6.24*sin(3*S)+N) / Arcs;
  }

  double ILE_Pert::dist()
  {
    return R_Earth * Arcs / (sinPi * 0.999953253);
  }

}

//------------------------------------------------------------------------------
//
// MoonPos: вычисление эклиптических координат Луны с помощью теории Брауна
// Уточненные лунные эфемериды
//
// Входные данные:
//
//   T - время в юлианских столетиях, прошедших от эпохи J2000
//
// Возвращаемые параметры: геоцентрические координаты Луны (км), отнесенные к эклиптике
// и равноденствию даты
//
// Примечание: учитывается время распространения света
//
//------------------------------------------------------------------------------
Vec3D MoonPos (double T)
{
  //
  // Переменные
  //
  ILE_Pert Pert;

  Pert.Init(T);  // Инициализация

  // Возмущения от Солнца
  Pert.AddSol (   13.902,   14.06,-0.001,   0.2607,0, 0, 0, 4);
  Pert.AddSol (    0.403,   -4.01,+0.394,   0.0023,0, 0, 0, 3);
  Pert.AddSol ( 2369.912, 2373.36,+0.601,  28.2333,0, 0, 0, 2);
  Pert.AddSol ( -125.154, -112.79,-0.725,  -0.9781,0, 0, 0, 1);
  Pert.AddSol (    1.979,    6.98,-0.445,   0.0433,1, 0, 0, 4);
  Pert.AddSol (  191.953,  192.72,+0.029,   3.0861,1, 0, 0, 2);
  Pert.AddSol (   -8.466,  -13.51,+0.455,  -0.1093,1, 0, 0, 1);
```

```
Pert.AddSol (22639.500,22609.07,+0.079, 186.5398,1, 0, 0, 0);
Pert.AddSol (   18.609,     3.59,-0.094,   0.0118,1, 0, 0,-1);
Pert AddSol (-4586.465,-4578.13,-0.077,  34.3117,1, 0, 0,-2);
Pert.AddSol (   +3.215,     5.44,+0.192,  -0.0386,1, 0, 0,-3);
Pert.AddSol (  -38.428,   -38.64,+0.001,   0.6008,1, 0, 0,-4);
Pert.AddSol (   -0.393,    -1.43,-0.092,   0.0086,1, 0, 0,-6);
Pert.AddSol (   -0.289,    -1.59,+0.123,  -0.0053,0, 1, 0, 4);
Pert.AddSol (  -24.420,   -25.10,+0.040,  -0.3000,0, 1, 0, 2);
Pert.AddSol (   18.023,    17.93,+0.007,   0.1494,0, 1, 0, 1);
Pert.AddSol ( -668.146,  -126.98,-1.302,  -0.3997,0, 1, 0, 0);
Pert.AddSol (    0.560,     0.32,-0.001,  -0.0037,0, 1, 0,-1);
Pert AddSol ( -165.145,  -165.06,+0.054,   1.9178,0, 1, 0,-2);
Pert.AddSol (   -1.877,    -6.46,-0.416,   0.0339,0, 1, 0,-4).
Pert.AddSol (    0.213,     1.02,-0.074,   0.0054,2, 0, 0, 4);
Pert.AddSol (   14.387,    14.78,-0.017,   0.2833,2, 0, 0, 2);
Pert.AddSol (   -0.586,    -1.20,+0.054,  -0.0100,2, 0, 0, 1);
Pert.AddSol (  769.016,   767.96,+0.107,  10.1657,2, 0, 0, 0);
Pert.AddSol (   +1.750,     2.01,-0.018,   0.0155,2, 0, 0,-1);
Pert.AddSol ( -211.656,  -152.53,+5.679,  -0.3039,2, 0, 0,-2);
Pert.AddSol (   +1 225,     0.91,-0.030,  -0.0088,2, 0, 0,-3);
Pert.AddSol (  -30.773,   -34.07,-0.308,   0.3722,2, 0, 0,-4);
Pert.AddSol (   -0.570,    -1.40,-0.074,   0.0109,2, 0, 0,-6);
Pert.AddSol (   -2.921,   -11.75,+0.787,  -0.0484,1, 1, 0, 2);
Pert.AddSol (   +1.267,     1.52,-0.022,   0.0164,1, 1, 0, 1);
Pert.AddSol ( -109.673,  -115.18,+0.461,  -0.9490,1, 1, 0, 0);
Pert.AddSol ( -205.962,  -182.36,+2.056,  +1.4437,1, 1, 0,-2);
Pert.AddSol (    0.233,     0.36, 0.012,  -0.0025,1, 1, 0,-3);
Pert.AddSol (   -4.391,    -9.66,-0 471,   0.0673,1, 1, 0,-4);
Pert.AddSol (    0.283,     1.53,-0.111,  +0.0060,1,-1, 0,+4);
Pert.AddSol (   14.577,    31.70,-1.540,  +0.2302,1,-1, 0, 2);
Pert.AddSol (  147.687,   138.76,+0.679,  +1.1528,1,-1, 0, 0);
Pert AddSol (   -1.089,     0.55,+0.021,   0.0    ,1,-1, 0,-1);
Pert.AddSol (   28.475,    23.59,-0.443,  -0.2257,1,-1, 0,-2);
Pert.AddSol (   -0.276,    -0.38,-0.006,  -0.0036,1,-1, 0,-3);
Pert.AddSol (    0.636,     2.27,+0.146,  -0.0102,1,-1, 0,-4);
Pert.AddSol (   -0.189,    -1.68,+0.131,  -0.0028,0, 2, 0, 2);
Pert.AddSol (   -7.486,    -0.66,-0.037,  -0.0086,0, 2, 0, 0);
Pert.AddSol (   -8.096,   -16.35,-0.740,   0.0918,0, 2, 0,-2);
Pert.AddSol (   -5.741,    -0.04, 0.0  ,  -0.0009,0, 0, 2, 2);
Pert.AddSol (    0.255,     0.0 , 0.0  ,   0.0    ,0, 0, 2, 1);
Pert.AddSol ( -411.608,    -0.20, 0.0  ,  -0.0124,0, 0, 2, 0);
Pert.AddSol (    0.584,     0.84, 0.0  ,  +0.0071,0, 0, 2,-1);
Pert.AddSol (  -55.173,   -52.14, 0.0  ,  -0.1052,0, 0, 2,-2);
Pert.AddSol (    0.254,     0.25, 0.0  ,  -0.0017,0, 0, 2,-3);
Pert.AddSol (   +0.025,    -1.67, 0.0  ,  +0.0031,0, 0, 2,-4);
Pert.AddSol (    1.060,     2.96,-0.166,   0.0243,3, 0, 0,+2);
Pert.AddSol (   36.124,    50.64,-1.300,   0.6215,3, 0, 0, 0);
Pert.AddSol (  -13.193,   -16.40,+0.258,  -0.1187,3, 0, 0,-2);
Pert.AddSol (   -1.187,    -0.74,+0.042,   0.0074,3, 0, 0,-4);
Pert.AddSol (   -0.293,    -0.31,-0.002,   0.0046,3, 0, 0,-6).
Pert.AddSol (   -0.290,    -1.45,+0.116,  -0.0051,2, 1, 0, 2);
Pert.AddSol (   -7.649,   -10.56,+0.259,  -0.1038,2, 1, 0, 0);
Pert.AddSol (   -8.627,    -7.59,+0.078,  -0.0192,2, 1, 0,-2);
Pert.AddSol (   -2.740,    -2.54,+0.022,   0.0324,2, 1, 0,-4);
Pert.AddSol (    1.181,     3.32,-0.212,   0.0213,2,-1, 0,+2);
Pert.AddSol (    9.703,    11.67,-0.151,   0.1268,2,-1, 0, 0);
Pert.AddSol (   -0.352,    -0.37,+0.001,  -0.0028,2,-1, 0,-1);
```

```
  Pert.AddSol (   -2.494,   -1.17,-0.003,  -0.0017,2,-1, 0,-2);
  Pert.AddSol (    0.360,    0.20,-0.012,  -0.0043,2,-1, 0,-4);
  Pert.AddSol (   -1.167,   -1.25,+0.008,  -0.0106,1, 2, 0, 0);
  Pert.AddSol (   -7.412,   -6.12,+0.117,   0.0484,1, 2, 0,-2);
  Pert.AddSol (   -0.311,   -0.65,-0.032,   0.0044,1, 2, 0,-4);
  Pert.AddSol (   +0.757,    1.82,-0.105,   0.0112,1,-2, 0, 2);
  Pert.AddSol (   +2.580,    2.32,+0.027,   0.0196,1,-2, 0, 0);
  Pert.AddSol (   +2.533,    2.40,-0.014,  -0.0212,1,-2, 0,-2);
  Pert.AddSol (   -0.344,   -0.57,-0.025,  +0.0036,0, 3, 0,-2);
  Pert.AddSol (   -0.992,   -0.02, 0.0  ,   0.0   ,1, 0, 2, 2);
  Pert.AddSol (  -45.099,   -0.02, 0.0  ,  -0.0010,1, 0, 2, 0);
  Pert.AddSol (   -0.179,   -9.52, 0.0  ,  -0.0833,1, 0, 2,-2);
  Pert.AddSol (   -0.301,   -0.33, 0.0  ,   0.0014,1, 0, 2,-4);
  Pert.AddSol (   -6.382,   -3.37, 0.0  ,  -0.0481,1, 0,-2, 2);
  Pert.AddSol (   39.528,   85.13, 0.0  ,  -0.7136,1, 0,-2, 0);
  Pert.AddSol (    9.366,    0.71, 0.0  ,  -0.0112,1, 0,-2,-2);
  Pert.AddSol (    0.202,    0.02, 0.0  ,   0.0   ,1, 0,-2,-4);
  Pert.AddSol (    0.415,    0.10, 0.0  ,   0.0013,0, 1, 2, 0);
  Pert.AddSol (   -2.152,   -2.26, 0.0  ,  -0.0066,0, 1, 2,-2);
  Pert.AddSol (   -1.440,   -1.30, 0.0  ,  +0.0014,0, 1,-2, 2);
  Pert.AddSol (    0.384,   -0.04, 0.0  ,   0.0   ,0, 1,-2,-2);
  Pert.AddSol (   +1.938,   +3.60,-0.145,  +0.0401,4, 0, 0, 0);
  Pert.AddSol (   -0.952,   -1.58,+0.052,  -0.0130,4, 0, 0,-2);
  Pert.AddSol (   -0.551,   -0.94,+0.032,  -0.0097,3, 1, 0, 0);
  Pert.AddSol (   -0.482,   -0.57,+0.005,  -0.0045,3, 1, 0,-2);
  Pert.AddSol (    0.681,    0.96,-0.026,   0.0115,3,-1, 0, 0);
  Pert.AddSol (   -0.297,   -0.27, 0.002,  -0.0009,2, 2, 0,-2);
  Pert.AddSol (    0.254,   +0.21,-0.003,   0.0   ,2,-2, 0,-2);
  Pert.AddSol (   -0.250,   -0.22, 0.004,   0.0014,1, 3, 0,-2);
  Pert.AddSol (   -3.996,    0.0 , 0.0  ,  +0.0004,2, 0, 2, 0);
  Pert.AddSol (    0.557,   -0.75, 0.0  ,  -0.0090,2, 0, 2,-2);
  Pert.AddSol (   -0.459,   -0.38, 0.0  ,  -0.0053,2, 0,-2, 2);
  Pert.AddSol (   -1.298,    0.74, 0.0  ,  +0.0004,2, 0,-2, 0);
  Pert.AddSol (    0.538,    1.14, 0.0  ,  -0.0141,2, 0,-2,-2);
  Pert.AddSol (    0.263,    0.02, 0.0  ,   0.0   ,1, 1, 2, 0);
  Pert.AddSol (    0.426,   +0.07, 0.0  ,  -0.0006,1, 1,-2,-2);
  Pert.AddSol (   -0.304,   +0.03, 0.0  ,  +0.0003,1,-1, 2, 0);
  Pert.AddSol (   -0.372,   -0.19, 0.0  ,  -0.0027,1,-1,-2, 2);
  Pert.AddSol (   +0.418,    0.0 , 0.0  ,   0.0   ,0, 0, 4, 0);
  Pert.AddSol (   -0.330,   -0.04, 0.0  ,   0.0   ,3, 0, 2, 0);

  // Возмущения от Солнца в широте
  Pert.AddN(-526.069, 0, 0,1,-2);   Pert.AddN(  -3.352, 0, 0,1,-4);
  Pert.AddN( +44.297,+1, 0,1,-2);   Pert.AddN(  -6.000,+1, 0,1,-4);
  Pert.AddN( +20.599,-1, 0,1, 0);   Pert.AddN( -30.598,-1, 0,1,-2);
  Pert.AddN( -24.649,-2, 0,1, 0);   Pert.AddN(  -2.000,-2, 0,1,-2);
  Pert.AddN( -22.571, 0,+1,1,-2);   Pert.AddN( +10.985, 0,-1,1,-2);

  // Планетные возмущения
  Pert.Planetary(T);

  // Позиционный вектор
  return Vec3D ( Polar(Pert.lambda(),Pert.beta(),Pert.dist()) );
}
```

Функция MoonPos может быть дополнена MoonEqu, вычисляющей истинные экваториальные координаты взамен средних эклиптических координат.

```
//------------------------------------------------------------------------
//
// MoonEqu: вычисление экваториальных координат Луны с помощью теории Брауна
// Уточненные лунные эфемериды
//
// Входные данные:
//
//   T - время в юлианских столетиях, прошедших от эпохи J2000
//
// Возвращаемые данные: геоцентрические экваториальные координаты Луны (км), отнесенные
// к истинному равноденствию даты
//
// Замечание: учитывается время распространения света
//
//------------------------------------------------------------------------
Vec3D MoonEqu (double T)
{
  return NutMatrix(T) * Ecl2EquMatrix(T) * MoonPos(T);
}
```

8.3. Приближение Чебышева

Несмотря на все хитрости, использованные для определения возмущений, вычисление координат Луны с помощью функции MoonPos требует относительно больших затрат времени и усилий. На практике это особенно заметно, когда требуется вычислить сразу большое число положений Луны. В одной из следующих глав мы зададимся целью предсказания покрытий звезд Луной, для чего мы используем итерационную процедуру, позволяющую установить моменты соединения Луны и избранной звезды. Чтобы подготовиться к эффективному решению столь сложной вычислительной задачи, мы сейчас усовершенствуем специальный класс C++, с помощью которого координаты тела могут быть аппроксимированы в удобной для практического использования форме. Это поможет нам сэкономить время, и когда потребуется определить большое число значений координат, мы прибегнем к аппроксимации.

Простой пример подобного приближения уже рассматривался в главе 3, когда мы описывали высоту Солнца и Луны с помощью параболы. Тогда для каждых трех точек функции мы определяли полином второго порядка, который приблизительно описывал поведение функции в пределах заданного интервала. Подобным образом мы можем взять n точек (x_i, y_i), где $i = 1...n$, и однозначно определить полином $P_n(x)$ порядка $n - 1$, который соответствует заданным точкам. Существует множество способов, чтобы убедиться в существовании такого полинома. Здесь мы упомянем лишь методы Лагранжа и Ньютона.

К сожалению, во многих случаях определенный таким способом полином оказывается непригодным на практике. Заданные значения функции представляются удовлетворительно, однако в промежутке между этими точками полином демонстрирует отклонения, совершенно не соответствующие кривой аппроксимируемой функции, и эти отклонения непредсказуемо возрастают у границ исследуемого интервала. Этот эффект тем значительнее, чем больше выбранный

порядок полинома. Поэтому пригодные результаты можно получить лишь методом интерполяции Лагранжа при низких порядках (примерно до $n = 5$). Нам требуется процедура, обеспечивающая гладкое приближение даже при использовании полиномов высоких порядков. Мы сейчас покажем, как такое приближение может быть получено. Ключом к решению задачи является так называемый полином Чебышева (рис. 8.1).

Полином Чебышева порядка n (для $|x| \le 1$) определяется как

$$T_n(x) = \cos(n \cdot \arccos x). \tag{8.8}$$

Из этого тригонометрического представления не следует со всей очевидностью, что мы имеем дело именно с полиномом. Это ясно видно лишь при $n = 0$ ($T_0 = 1$) и $n = 1$ ($T_1 = x$). Дополнительная теорема косинусов позволит нам легко вывести рекурсивное соотношение для T_n, из которого будет совершенно очевидно, что выражение (8.8) действительно задает полином. Из равенства

$$\cos(\alpha + \beta) + \cos(\alpha - \beta) = 2\cos\alpha\cos\beta$$

следует, что

$$\cos((n+1)\varphi) = 2\cos(n\varphi)\cos\varphi - \cos((n-1)\varphi). \tag{8.9}$$

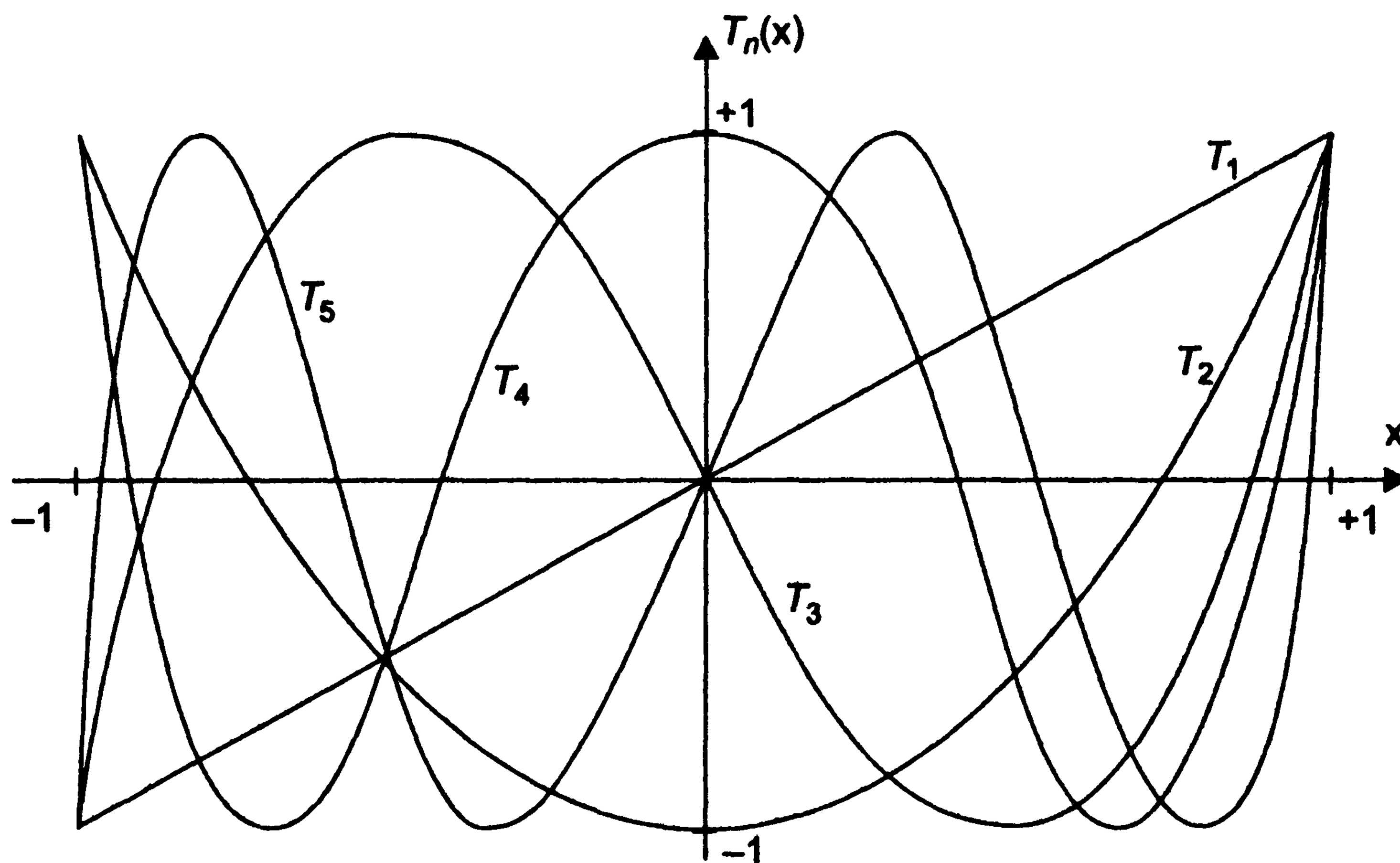


Рис. 8.1. Полиномы Чебышева от T_1 до T_5

Если $\varphi = \arccos x$, то после подстановки получим

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad \text{при} \quad n \ge 1. \tag{8.10}$$

Позднее мы детально рассмотрим это рекурсивное выражение. Однако в любом случае, теперь очевидно, что $T_n(x)$ по определению является полиномом порядка n. В полной форме первые полиномы Чебышева выглядят следующим образом:

$$T_0(x) = 1, \qquad T_3(x) = 4x^3 - 3x,$$
$$T_1(x) = x, \qquad T_4(x) = 8x^4 - 8x^2 + 1,$$
$$T_2(x) = 2x^2 - 1, \qquad T_5(x) = 16x^5 - 20x^3 + 5x.$$

Нам теперь необходимо рассмотреть более детально особенности полиномов Чебышева в интервале $[-1, +1]$. Легко убедиться, сделав подстановку в определение (8.8), что внутри этого интервала $T_n(x)$ обращается в нуль именно в n точках:

$$T_n(x) = 0 \quad \text{при} \quad x = \cos(\pi \cdot (k + 1/2)/n), \qquad \text{где} \quad k = 0, \ldots, n-1. \tag{8.11}$$

При $|x| \le 1$ значение $|T_n(x)|$ не может превышать 1.

Чтобы аппроксимировать произвольную функцию $f(x)$ в интервале $[a, b]$, мы заменим независимую переменную x нормированной переменной $\hat{x}$ в интервале $[-1, +1]$. Для этого потребуются преобразования

$$\hat{x} = \frac{x - \frac{1}{2}(a+b)}{\frac{1}{2}(b-a)} \quad \text{при} \quad x \in [a, b] \to \hat{x} \in [-1, +1]$$

и

$$x = \hat{x} \cdot \frac{1}{2}(b-a) + \frac{1}{2}(a+b) \quad \text{при} \quad \hat{x} \in [-1, +1] \to x \in [a, b].$$

Функция $f(x)$ теперь может быть представлена полиномами Чебышева вплоть до n-го порядка в виде

$$f(x) \approx f^*(x) = \sum_{j=0}^{n} c_j T_j(\hat{x}) - c_0/2. \tag{8.12}$$

Коэффициенты c_j в этой сумме вычисляются следующим образом:

$$c_j = \frac{2}{n+1} \sum_{k=0}^{n} f(x_k^{n+1}) T_j(\hat{x}_k^{n+1}), \tag{8.13}$$

где $\hat{x}_k^{n+1}$ представляет k-й корень T_{n+1}. В полной форме отдельные члены этого уравнения в соответствии с (8.8) и (8.11) выглядят следующим образом:

$$x_k^{n+1} = \frac{(b-a)}{2} \cdot \cos\left(\pi \frac{2k+1}{2n+2}\right) + \frac{(a+b)}{2}, \qquad k = 0, 1, \ldots, n$$

$$T_j(\hat{x}_k^{n+1}) = \cos\left(j\pi \frac{2k+1}{2n+2}\right).$$

Можно видеть, что для вычисления коэффициентов c_j данную функцию f следует оценить в $(n + 1)$ точках. Эти точки не произвольные, как в интерполяции Лагранжа, а совершенно определенные: в них T_{n+1} обращается в нуль. В этом состоит секрет успеха приближения Чебышева. Функция f^* не просто полином

n-го порядка, совпадающий с функцией f в $(n + 1)$ точках $(f^*(x_k^{n+1}) = f(x_k^{n+1}))$. Поскольку выбранные точки у границ интервала располагаются теснее, нежели в его центре, общая ошибка аппроксимации сглаживается. Это, в частности, можно заметить, если при вычислении f^* суммировать не все члены в выражении (8.12). Если пренебречь наивысшим членом $c_n T_n$, то определенно можно утверждать, что возникшая ошибка будет менее $|c_n|$, поскольку $|T_n| \leq 1$ для всех значений в промежутке $[a, b]$. Подобная оценка невозможна для других итерационных процедур.

Разумеется, приближение Чебышева может использоваться в таком виде, лишь если возможно вычислить требуемые значения функции для всех заданных точек. Однако перед нами стоит задача аппроксимации координат небесных тел, которые в принципе можно узнать для любого произвольного момента времени. Поэтому указанное ограничение не является препятствием для использования приближения Чебышева в наших целях.

Ниже мы полагаем, что имеем дело с функцией от времени, которая задает трехмерный вектор типа Vec3D:

```
// Прототип функции Vec3D f(double t)
typedef Vec3D (*C3Dfunct)(double t);
```

Примерами таких функций являются функции MoonPos и MoonEqu, а также функции для вычисления гелиоцентрических координат планет, описанные в главе 6. Для функций этого типа приближения Чебышева могут быть вычислены и оценены с помощью класса Cheb3D

```
// Cheb3D: трехмерное приближение Чебышева
class Cheb3D
{

  public:

    // Конструктор
    Cheb3D (
      C3Dfunct f,    // Указатель на используемую функцию
      int n,         // Степень многочлена
      double dt      // Размер интервала
    );

    // Деструктор
    ~Cheb3D();

    // Fit: аппроксимирование функции m_f в интервале [ta, tb]
    void Fit (double ta, double tb);

    // Оценка приближения по аргументу t, при необходимости - повтор вычисления функции Fit
    Vec3D Value (double t);

  private:

    // Члены

    C3Dfunct m_f;            // Функция
    int      m_n;            // Степень
```

```
    bool      m_Valid;          // Флаг проверки допустимости коэффициентов Чебышева
    double    m_dt;             // Размер интервала
    double    m_ta,m_tb;        // Интервал
    double    *Cx,*Cy,*Cz;      // Коэффициенты Чебышева

};
```

Конструктор

```
Cheb3D::Cheb3D (C3Dfunct f, int n, double dt)
  : m_n(n), m_f(f), m_dt(dt), m_Valid(false)
{
  Cx = new double[n+1];
  Cy = new double[n+1];
  Cz = new double[n+1];
};
```

вначале выделяет необходимую память для коэффициентов полинома Cx, Cy и Cz при x-, y- и z-координатах. Дополнительно степень полинома, желаемый размер интервала и указатель на функцию приближения помещаются в соответствующие переменные. Наконец, метод Fit применяется для вычисления коэффициентов полинома Чебышева для всех трех координат в пределах заранее определенного промежутка $[t_a, t_b]$.

```
//
// Fit: аппроксимирование функции m_f в интервале [ta, tb]
//
void Cheb3D::Fit (double ta, double tb)
{
  int     j, k;
  double  tau, t, fac;
  double* T=new double[m_n+1];
  Vec3D   f;

  // Очистка всех значений коэффициентов
  for (j=0; j<=m_n; j++) Cx[j]=Cy[j]=Cz[j]=0.0;

  // Цикл нахождения корней T^(n+1)
  for (k=0; k<=m_n; k++) {

    tau = cos((2*k+1)*pi/(2*m_n+2));         // Опорная точка tau_k
    t = ((tb-ta)/2.0) * tau + ((tb+ta)/2.0); // Время t
    f = m_f(t);                              // Значение функции в момент времени t

    // Вычисление коэффициентов C_j с использованием рекуррентного соотношения
    for (j=0; j<=m_n; j++) {
      switch (j) {
        case  0: T[j]=1.0; break;
        case  1: T[j]=tau; break;
        default: T[j]=2.0*tau*T[j-1]-T[j-2];
      };
      // Приращение коэффициента C_j на f(t)*T_j(tau)
      Cx[j]+=f[x]*T[j]; Cy[j]+=f[y]*T[j]; Cz[j]+=f[z]*T[j];
    };
  };

  // Масштабирование
```

```
  fac = 2.0/(m_n+1);
  for (j=0; j<=m_n, j++) { Cx[j]*=fac; Cy[j]*=fac; Cz[j]*=fac; }

  // Коррекция некоторых членов
  m_ta = ta,
  m_tb = tb,
  m_Valid = true;

  delete[] T;
}
```

В рамках этого метода вычисляются значения функции $f(t)$ и — ввиду рекурсии — значения $T_j(\tau)$ полиномов Чебышева порядков $j = 0,..., n$ для отдельных корней $\tau = \hat{x}_k^{n+1}$ полинома Чебышева порядка $n + 1$. Искомые коэффициенты выводятся отсюда в соответствии с (8.13).

Оценивая заданное разложение с помощью полиномов Чебышева, нет необходимости производить точное вычисление полиномов. Алгоритм Кленшоу (Clenshaw) дает правило для оценки разложения Чебышева (8.12) порядка n с коэффициентами ($c_0, c_1,..., c_n$) при обращении к рекурсивному отношению из (8.10):

- Установить $f_{n+1} = 0$ и $f_{n+2} = 0$.
- Вычислить ряды с нормированным аргументом $\hat{x}$.

$$f_i = 2\hat{x}\, f_{i+1} - f_{i+2} + c_i \quad \text{при} \quad i = n, n-1, ..., 0.$$

- Требуемое значение функции:

$$f(x) = (f_0 - f_2)/2 = \hat{x} f_1 - f_2 + c_0/2.$$

Метод Value оценивает разложение Чебышева следующим образом:

```
// Value: оценка приближения по аргументу t
//
Vec3D Cheb3D::Value (double t)
{
  //
  // Константы
  //
  const double eps=0.01; // Частичное перекрытие

  //
  // Переменные
  //
  Vec3D  f1, f2, old_f1;
  double tau, k;

  // Создание новых коэффициентов
  if ( !m_Valid || (t<m_ta) || (m_tb<t) ) {
    k = floor(t/m_dt);
    Fit ( (k-eps)*m_dt, (k+1+eps)*m_dt );
  }

  // Оценка приближения
  tau = (2 0*t-m_ta-m_tb)/(m_tb-m_ta);

  for (int i=m_n; i>=1; i--) {
    old_f1 = f1;
```

```
    f1 = 2.0*tau*f1-f2+Vec3D(Cx[i].Cy[i].Cz[i]);
    f2 = old_f1;
  };

  return tau*f1-f2+0.5*Vec3D(Cx[0].Cy[0].Cz[0]);
}
```

8.4. Программа Luna

Программа Luna объединяет различные процедуры, описанные в настоящей главе, в одно небольшое приложение. Она может использоваться для вычисления лунных эфемерид, то есть таблиц координат Луны, как они обычно представлены во многих календарях. Программа выдает видимые экваториальные координаты Луны (отнесенные к равноденствию даты), ее расстояние в радиусах Земли и горизонтальный параллакс. Программа производит разложение лунных координат в виде полиномов Чебышева с временным интервалом в десять дней. Это разложение в ряд может быть легко и быстро оценено. Одиннадцать положений Луны вычисляются таким образом, что приближение с помощью этих рядов может быть получено с требуемой точностью. Следовательно, если нам требуются координаты Луны с интервалом в сутки или около того, потребуются дополнительные вычисления. Составление эфемерид с еще меньшим интервалом, как это требуется, например, в навигации, связано со значительными затратами времени.

```
//------------------------------------------------------------------------------
// Файл:     Luna.cpp
// Назначение программы: расчет лунных эфемерид
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//------------------------------------------------------------------------------

#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Cheb.h"
#include "APC_Const.h"
#include "APC_Math.h"
#include "APC_Moon.h"
#include "APC_Time.h"
#include "APC_VecMat3D h"

#ifdef __GNUC__  // GNU C++ adaptation
#include "GNU_iomanip h"
#endif

using namespace std;

//------------------------------------------------------------------------------
//
// GetEph - запрос пользователю о датах начала и окончания эфемерид, а также о размере шага
//
// Output:
```

```
//
//   MjdStart - модифицированная юлианская дата начала эфемерид
//   Step - размер шага эфемерид в сутках
//   MjdEnd - модифицированная юлианская дата окончания эфемерид
//
//------------------------------------------------------------------------
void GetEph (double& MjdStart, double& Step, double& MjdEnd)
{
  //
  // Переменные
  //
  int    year, month, day;
  double hour;

  cout << endl
       << " Begin and end of the ephemeris: " << endl
       << endl;

  cout << "  first date (yyyy mm dd hh.hhh) ..  ";
  cin >> year >> month >> day >> hour, cin.ignore(81,'\n');
  MjdStart = Mjd(year,month,day) + hour/24.0 ;

  cout << "  final date (yyyy mm dd hh.hhh) . . ";
  cin >> year >> month >> day >> hour; cin.ignore(81,'\n');
  MjdEnd = Mjd(year,month,day) + hour/24.0 ;

  cout << "  step size (dd hh.hh)           ... ";
  cin >> day >> hour; cin.ignore(81,'\n');
  Step  = day + hour/24.0;
}

//------------------------------------------------------------------------
//
// Основная программа
//
//------------------------------------------------------------------------
void main()
{
  //
  // Константы
  //
  const int    Degree    = 10;
  const double Interval  = 10.0/36525.0;  // 10d in [cy]

  //
  // Переменные
  //
  int    n_line = 0;
  double MjdStart, Step, MjdEnd;
  double Date, T;
  double Dist, Parallax;
  Vec3D  r_Moon;
  Cheb3D ChebMoonEqu (MoonEqu, Degree, Interval);

  // Title
  cout << endl
       << "                         LUNA: lunar ephemeris           " << endl
```

```
      << "        (c) 1999 Oliver Montenbruck, Thomas Pfleger  " << endl
      << endl;

  // Приглашение пользователю ввести начальную дату, величину шага и конечную дату
  GetEph (MjdStart, Step, MjdEnd);

  // Заголовок
  cout << endl << endl
      << "    Date      ET        RA             Dec      Distance  "
      << " Parallax " << endl
      << "                        h  m  s       o  '  \"  Earth radii"
      << "    '  \"    " << endl;

  // Вычисление эфемерид
  Date = MjdStart;

  // Цикл по времени
  while ( Date < MjdEnd + Step/2 ) {

    // Геоцентрические кординаты луны, полученные приближением полиномами Чебышева
    T = ( Date - MJD_J2000 ) / 36525.0;

    r_Moon = ChebMoonEqu.Value(T);

    // Расстояние и параллакс
    Dist     = r_Moon[r];
    Parallax = asin(R_Earth/Dist);

    // Вывод
    cout << " " << DateTime(Date,HHMM)
         << fixed
         << setprecision(2) << setw(14) << Angle(Deg*r_Moon[phi]/15.0,DMMSSs)
         << "  " << showpos
         << setprecision(1) << setw(11) << Angle(Deg*r_Moon[theta],DMMSSs)
         << noshowpos
         << setprecision(3) << setw(10) << Dist/R_Earth
         << setprecision(2) << setw(12) << Angle(60.0*Deg*Parallax,DMMm)
         << endl;

    ++n_line; if ( (n_line % 5) ==0 ) cout << endl;

    Date += Step; // Следующий шаг

  }; // Конец цикла

}
```

Чтобы проиллюстрировать работу программы, вычислим лунные эфемериды с шагом в двое суток для января 1989 года. Для работы программы необходимо ввести лишь граничные даты эфемерид в виде: год, месяц, число и часы; а также величину шага в форме: сутки и часы. Необходимо отметить, что, как и в случае вычисления положений планет, моменты времени указываются по эфемеридному времени. Данные, введенные пользователем, выделены курсивом.

```
LUNA: lunar ephemeris
      (c) 1999 Oliver Montenbruck, Thomas Pfleger
```

```
Begin and end of the ephemeris:

 first date (yyyy mm dd hh.hhh) ... 1989 01 01 00.0
 final date (yyyy mm dd hh.hhh) ... 1989 01 31 00.0
 step size (dd hh.hh)           ... 2 00.0
   Date       ET        RA           Dec       Distance    Parallax
                      h  m  s      o  '  "   Earth radii     '  "
1989/01/01 00:00   13 05 26.21  -10 42 58.7    63.053     54 31.41
1989/01/03 00:00   14 38 15.31  -20 23 19.3    61.961     55 29.10
1989/01/05 00·00   16 26 11.87  -26 51 42.9    60.407     56 54 71
1989/01/07 00:00   18 27 52.77  -27 43 03.3    58.861     58 24.44
1989/01/09 00:00   20 29 55.18  -21 44 30.5    57.789     59 29.44

1989/01/11 00:00   22 21 11.90  -10 29 51.2    57.442     59 51.00
1989/01/13 00.00    0 03 44.05  + 2 54 49.4    57.757     59 31.44
1989/01/15 00.00    1 46 21.72  +15 30 04.5    58.481     58 47.18
1989/01/17 00:00    3 36 54.09  +24 39 14.7    59.389     57 53.29
1989/01/19 00:00    5 35 18.56  +28 13 06.1    60.375     56 56 57

1989/01/21 00.00    7 31 01.72  +25 29 44.7    61.406     55 59.16
1989/01/23 00:00    9 14 03.91  +17 51 26 1    62.408     55 05.27
1989/01/25 00·00   10 44 27.57  + 7 31 36.2    63.194     54 24.13
1989/01/27 00·00   12 08 35.18  - 3 38 59.4    63.512     54 07.79
1989/01/29 00.00   13 34 26.38  -14 17 56.9    63 141     54 26.88

1989/01/31 00 00   15 10 02.69  -23 01 46.5    62 014     55 26.26
```

Что поразительно в этом примере, так это высокое значение, которого склонение Луны достигало примерно 7 и 19 января 1989 года. В этом году линия узлов лунной орбиты располагалась таким образом, что наклон эклиптики (23°,5) и наклонение орбиты луны (5°,1) складывались с одним знаком, поэтому склонение могло превосходить 28°. Следовательно, в определенные моменты времени этого года Луна находилась или очень высоко, или очень низко над горизонтом, а азимуты точек восхода и захода достигали экстремальных значений.

9. Солнечные затмения

Примерно раз в тридцать дней, в новолуние Луна обращена к Земле своей неосвещенной стороной. Наблюдатель, находящийся над плоскостью эклиптики, обнаружил бы, что Солнце, Луна и Земля выстроились в линию. Однако тень, отбрасываемая Луной, редко касается Земли. Поскольку лунная орбита наклонена к эклиптике, Луна в новолунии обычно оказывается или выше, или ниже ее. Луна пересекает эклиптику дважды в месяц, и только в том случае, когда это событие совпадает с новолунием, Солнце, Луна и Земля действительно оказываются на одной прямой и лунная тень падает на участок земной поверхности. Тем не менее солнечные затмения происходят дважды в год, но лишь немногочисленные счастливцы, оказавшиеся в узкой полосе, прочерчиваемой лунной тенью, имеющей на Земле поперечник около 100 км, имеют возможность видеть полное затмение Солнца.

Если бы плоскость лунной орбиты сохраняла неизменное положение в пространстве, все затмения происходили бы в два строго определенных месяца. Движение линии узлов приводит к тому, что даты затмений сдвигаются за год в среднем на три недели. Период полного обращения линии узлов, составляющий 18,6 лет, непосредственно определяет и картину наступления солнечных затмений (рис. 9.1).

9.1. Фазы Луны и затмения

Чтобы определить, возможно ли наступление солнечного затмения в течение года, необходимо сперва вычислить даты новолуний каждого месяца и затем — положение Луны относительно эклиптики. Новолуние наступает в тот момент, когда эклиптическая долгота Солнца и Луны ($\lambda_\odot$ и λ_M соответственно) совпадают, иными словами, когда разность $\lambda_M - \lambda_\odot$ равна нулю. В первой четверти она составляет 90° ($\pi/2$), в полнолуние — 180° (π) и 270° ($3\pi/2$) в последней четверти.

Определение фаз Луны, таким образом, представляет собой задачу нахождения точек, в которых обращается в нуль функция

$$\Delta\lambda = \lambda_M - \lambda_\odot - k\cdot\pi/2, \tag{9.1}$$

где $k = 0$ для новолуния, $k = 1$ для первой четверти, $k = 2$ для полнолуния и $k = 3$ для последней четверти. При необходимости следует прибавить или отнять число, кратное 2π, поскольку результат должен быть заключен в интервале $[-\pi, +\pi]$. В приведенной ниже программе PhasesFunc для вычисления эклиптической долготы Солнца и Луны используются функции SunPos и MoonPos. При вычислении

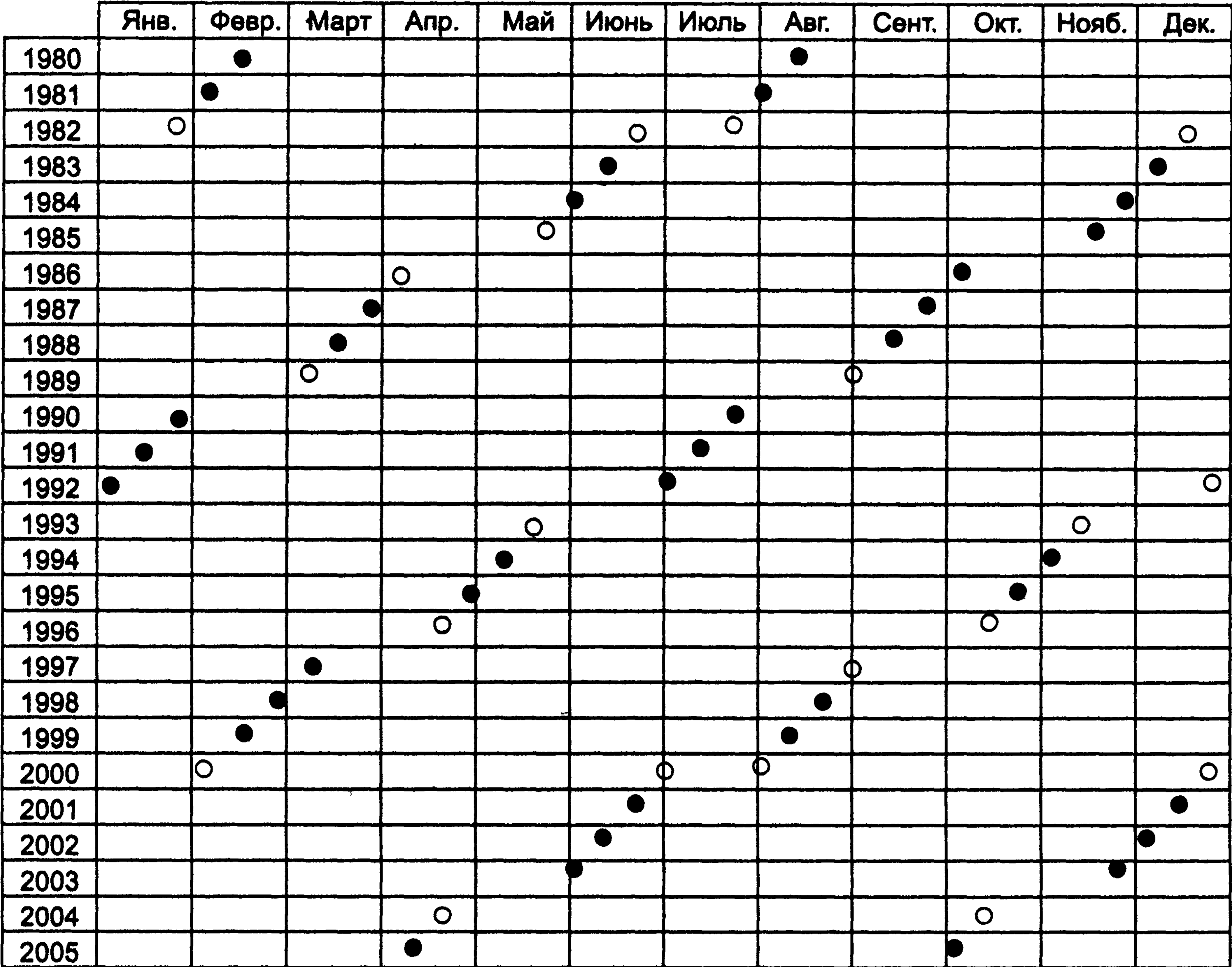


Рис. 9.1. Солнечные затмения в период 1980–2005 гг. (● — полное или кольцевое, ○ — частное)

долготы Солнца учитывается также поправка на время распространения света, составляющая примерно восемь минут.

```
enum enLunarPhase { NewMoon, FirstQuarter, FullMoon, LastQuarter };
enLunarPhase Phase = NewMoon;
//------------------------------------------------------------------------
//
// PhasesFunc: Функция для определения фаз Луны
//   T         эфемеридное время (в юлианских столетиях, прошедших после J2000)
// Возвращаемые значения:  разность долгот Солнца и Луны
// и величина фазы в радианах (0 в новолунии, pi/2 в первой четверти и т.д.)
//------------------------------------------------------------------------
double PhasesFunc (double T)
{
  const double tau_Sun  = 8.32 / (1440.0*36525.0);     // 8.32 min  [cy]

  const double LongDiff = MoonPos(T)[phi] - SunPos(T-tau_Sun)[phi];

  return  Modulo ( LongDiff - Phase*pi/2.0 + pi, pi2 ) - pi;
}
```

Сперва производится поиск с шагом в семь дней, в результате которого определяется интервал времени $[t_0\ , t_1]$, когда величина $\Delta\lambda$ меняет знак на положи-

тельный. После этого точное значение момента времени, когда эта величина обращается в нуль, находится методом итерации с использованием хорошо известных численных методов решения. В программе Phases использован так называемый Pegasus-метод, относящийся к типу метода regula falsi, но обеспечивающий лучшую сходимость (см. раздел 10.2).

Таблица 9.1. Критерии вероятности наступления солнечных и лунных затмений в зависимости от эклиптической широты Луны в новолуние или полнолуние, используемые программой Phases

Новолуние		
$\mid\beta\mid < 0°52'20''$	Центральное солнечное затмение	c
$\mid\beta\mid < 1°02'36$	Центральное солнечное затмение возможно	c?
$\mid\beta\mid < 1°24'33''$	Частное солнечное затмение	p
$\mid\beta\mid < 1°34'50''$	Частное солнечное затмение возможно	p?
$\mid\beta\mid > 1°34'50''$	Затмение невозможно	
Полнолуние		
$\mid\beta\mid < 0°21'50''$	Полное лунное затмение	t
$\mid\beta\mid < 0°32'14''$	Полное лунное затмение возможно	t?
$\mid\beta\mid < 0°53'24''$	Частное лунное затмение	p
$\mid\beta\mid < 1°03'50''$	Частное лунное затмение возможно	p?
$\mid\beta\mid < 1°26'15$	Полутеневое лунное затмение	P
$\mid\beta\mid < 1°36'43''$	Полутеневое лунное затмение возможно	P?
$\mid\beta\mid > 1°36'43$	Затмение невозможно	

Значение эклиптической широты β Луны в новолуние определяет возможность наступления солнечного затмения, а в полнолуние — лунного. Соответственно величине наклонения лунной орбиты к эклиптике β изменяется в течение года от $-5°$ до $+5°$. Солнечные затмения происходят в те два новолуния, когда эклиптическая широта Луны минимальна, однако в течение года может быть и более двух затмений. Примером тому являются годы 1982 и 2000, в которые произошло по четыре частных затмения. Полное солнечное затмение может наступить при значении эклиптической широты Луны, не превосходящем одного градуса, частные затмения еще возможны при $\beta \approx 1\overset{\circ}{.}5$. Можно также узнать кое-что об условиях видимости затмения. Если, например, Луна находится к северу от эклиптики ($\beta > 0$), область лунной тени будет лежать в северном полушарии Земли.

Программа Phases, которая не приводится здесь ради экономии места, выполняет все необходимые действия. После введения избранного года вычисляются моменты соответствующих лунных фаз и по критериям табл. 9.1 определяется возможность наступления солнечных или лунных затмений. Пример использования этой программы приведен в конце настоящей главы.

9.2. Геометрия затмений

Тень, отбрасываемая Луной, состоит из двух конических областей, известных под названием тени и полутени (рис. 9.2). Для наблюдателя, находящегося внутри тени, Луна имеет большие видимые размеры, чем Солнце, и полностью его заслоняет. Вершина конуса лунной тени находится на расстоянии около 375 000 км от Луны. На больших расстояниях наблюдается кольцевое затмение, поскольку диск Луны не может полностью закрыть Солнце и часть его поверхности остается видимой как яркое кольцо. Такое затмение наблюдалось, например, 29 апреля 1976 года в Средиземном море. Внутри полутени Луна заслоняет Солнце лишь частично, и последнее предстает в виде полумесяца большей или меньшей ширины. Фаза частного затмения тем больше, чем ближе наблюдатель находится к области тени.

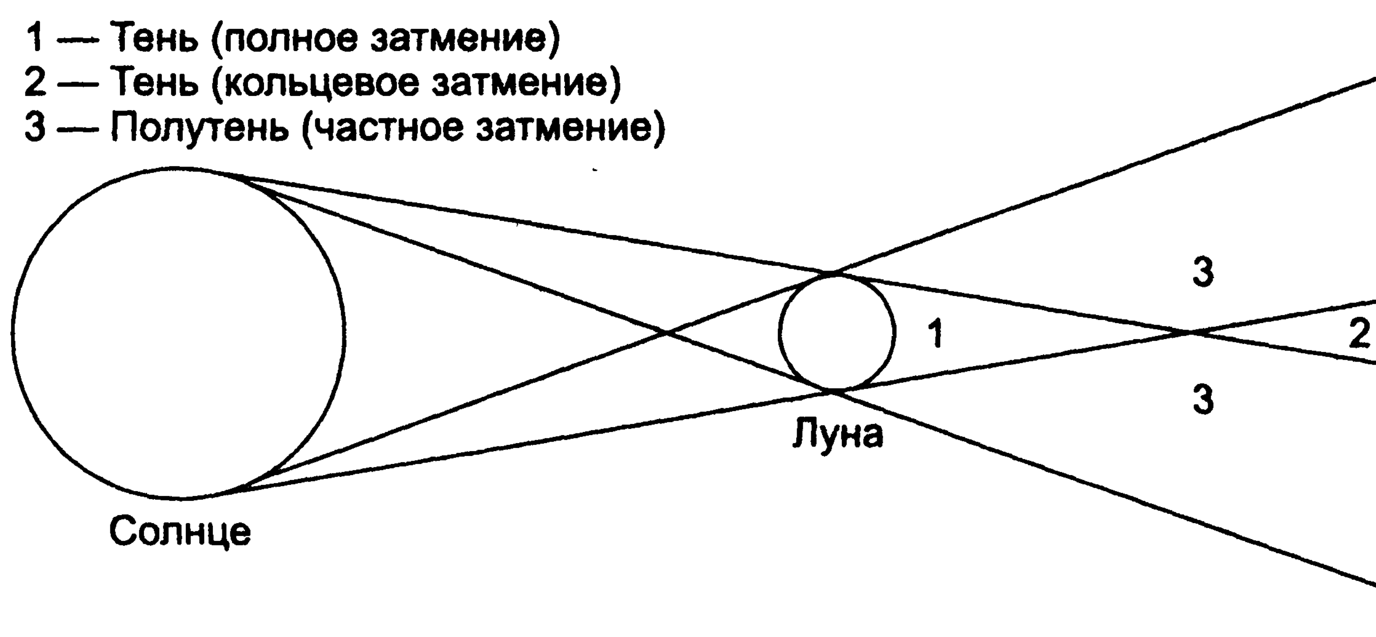


Рис. 9.2. Тень и полутень

Среднее расстояние от Земли до Луны составляет 380 000 км. Эксцентричность лунной орбиты обусловливает отклонения от этой величины примерно на 25 000 км. Таким образом, во время затмений вершина конуса лунной тени всегда находится близко к Земле, и даже когда Луна находится в перигее, диаметр ее тени на поверхности Земли редко превосходит 200 км. Если Луна в апогее, происходит кольцевое затмение. В предельном случае может наблюдаться кратковременное полное затмение, сменяющееся кольцевым, как это произошло 29 марта 1987 года.

Диаметры лунной тени (d) и полутени (D) на расстоянии s от Луны вычисляются по формулам:

$$d(s) = D_{\odot}\left(\frac{s}{r_{\odot M}}\right) - D_M\left(1 + \frac{s}{r_{\odot M}}\right)$$

и

$$D(s) = D_{\odot}\left(\frac{s}{r_{\odot M}}\right) + D_M\left(1 + \frac{s}{r_{\odot M}}\right). \qquad (9.2)$$

Здесь $D_{\odot}$ и D_M — диаметры Солнца и Луны соответственно:

$$D_{\odot} = 1\,392\,000 \text{ км} = 218{,}25\ R_{\oplus},$$

$$D_M = 3474 \text{ км} = 0{,}5450\ R_{\oplus},$$

а $r_{\odot M}$ — расстояние от Солнца до Луны.

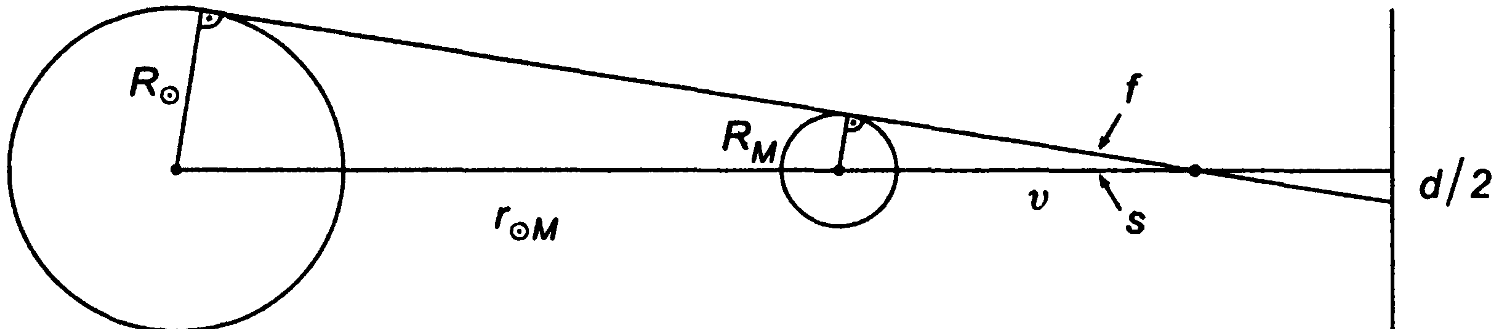


Рис. 9.3. Вычисление диаметра тени

Вывод этих формул легко понять из рис. 9.3. Здесь f — половина значения угла при вершине конуса тени, а v — расстояние между вершиной конуса и центром Луны. Обозначим радиусы Солнца и Луны $R_{\odot}$ и R_M соответственно, в таком случае

$$\sin(f)\cdot v = R_M \quad \text{и} \quad \sin(f)\cdot(v + r_{\odot M}) = R_{\odot}$$

или после преобразований

$$\sin(f) = \frac{R_{\odot} - R_M}{r_{\odot M}} \quad \text{и} \quad v = \frac{R_M \cdot r_{\odot M}}{R_{\odot} - R_M}.$$

Диаметр тени на расстоянии s от Луны, таким образом, равен

$$\begin{aligned} d &= 2(s - v)\cdot \operatorname{tg}(f) \\ &\approx 2(s - v)\cdot \sin(f) \\ &= 2R_{\odot}\left(\frac{s}{r_{\odot M}}\right) - 2R_M\left(1 + \frac{s}{r_{\odot M}}\right). \end{aligned}$$

Формула для диаметра полутени D может быть получена подобным же образом. Знак при d в формулах выше выбран соответственно ситуации реальной тени: d отрицательно при полном затмении и положительно при частном. Если диаметр полутени вблизи Земли составляет примерно половину диаметра последней, диаметр d тени колеблется в пределах от $-$ 0,04 $R_{\oplus}$ (250 км) до + 0,06 $R_{\oplus}$ (350 км).

Чтобы определить положение лунной тени на Земле, нужно провести прямую через центры Солнца и Луны и найти точку ее пересечения с земной поверхностью (рис. 9.4). Направление этой прямой описывается единичным вектором

$$\boldsymbol{e} = \frac{\boldsymbol{r}_M - \boldsymbol{r}_{\odot}}{|\boldsymbol{r}_M - \boldsymbol{r}_{\odot}|}, \tag{9.3}$$

который может быть получен из координат $\boldsymbol{r}_{\odot}$ и $\boldsymbol{r}_M$ Солнца и Луны. Для точки тени мы имеем

$$\boldsymbol{r} = \boldsymbol{r}_M + s\boldsymbol{e} \tag{9.4}$$

или

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_M \\ y_M \\ z_M \end{pmatrix} + s \cdot \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix}$$

Здесь s — расстояние от точки до центра Луны. Конец вектора $\boldsymbol{r}$ лежит на поверхности Земли, и поскольку изначально мы предполагали, что она является правильной сферой радиусом $R_\oplus$, получим

$$\boldsymbol{r}^2 = x^2 + y^2 + z^2 = R_\oplus^2. \quad (9.5)$$

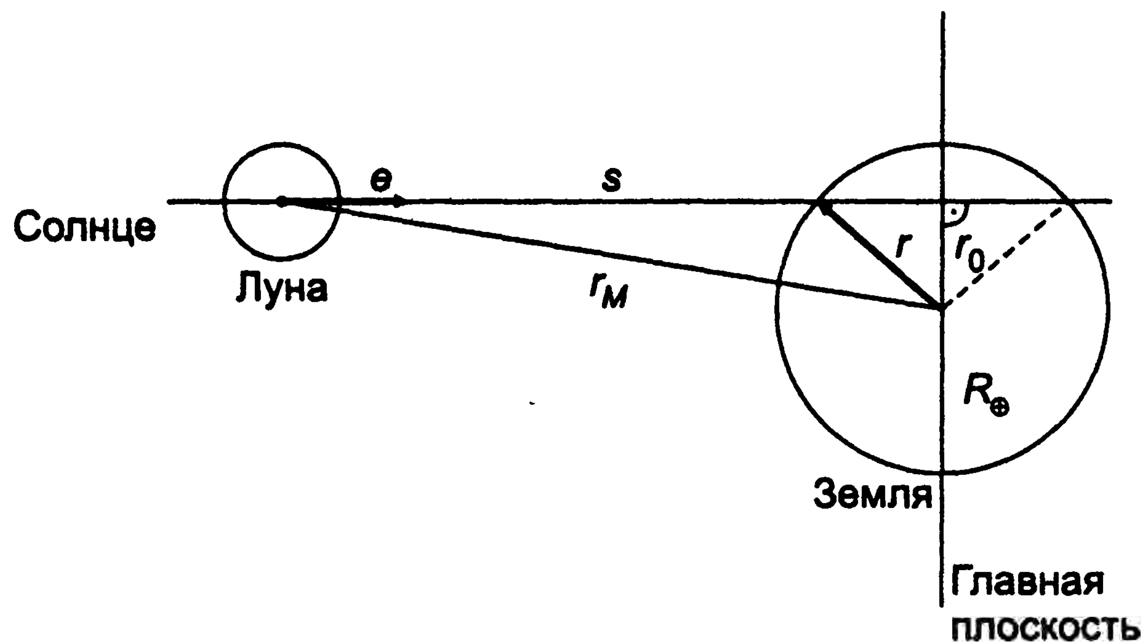


Рис. 9.4. Пересечение оси тени с Землей

Сделав подстановки, получаем квадратное уравнение

$$s^2 + 2(\boldsymbol{r}_M \cdot \boldsymbol{e})s + (r_M^2 - R_\oplus^2) = 0.$$

имеющее два решения

$$s = \begin{cases} s_0 - \sqrt{\Delta} & \text{для дневной стороны Земли,} \\ s_0 + \sqrt{\Delta} & \text{для ночной стороны Земли,} \end{cases}$$

где

$$s_0 = -\boldsymbol{r}_M \cdot \boldsymbol{e} = -(x_M e_x + y_M e_y + z_M e_z)$$

и

$$\Delta = s_0^2 + R_\oplus^2 \ r_M^2. \quad (9.6)$$

Понятно, что s_0 — это расстояние от Луны до главной плоскости, проходящей через центр Земли и перпендикулярной к оси лунной тени. Интересующая нас точка пересечения оси тени с поверхностью Земли лежит на расстоянии $\sqrt{\Delta}$ от этой плоскости на стороне, обращенной к Солнцу и Луне. Вторая точка пересечения находится на ночной стороне и не представляет интереса. Комбинируя уравнения, получим выражение для координат лунной тени на поверхности Земли

$$\boldsymbol{r} = \boldsymbol{r}_M + (s_0 - \sqrt{\Delta}) \cdot \boldsymbol{e}. \quad (9.7)$$

Естественно, мы берем положительное значение дискриминанта Δ, в противном случае лунная тень не касается Земли. Тем не менее солнечное затмение начинается задолго до полной фазы: как только полутень вступает на Землю, начинаются его частные фазы. Если расстояние r_0 между осью тени и центром Земли равно

$$r_0 = \sqrt{r_M^2 - s_0^2}$$

и d_0 и D_0 — диаметры лунной тени и полутени в главной плоскости, можно выработать критерий для определения доступных наблюдению фаз затмения:

$R_\oplus + D_0/2 < r_0$ — затмение не происходит;

$R_\oplus + |d_0|/2 < r_0 < R_\oplus + D_0/2$ — частная фаза;

$R_\oplus < r_0 < R_\oplus + |d_0|/2$ — нецентральное затмение;

$r_0 < R_\oplus$ — центральное затмение.

В случае нецентрального затмения конус лунной тени лишь чиркает по земной поверхности так, что ось тени не касается ее. Такое затмение, полное или кольцевое, видимо на очень небольшом участке Земли, и ни для какого земного наблюдателя Луна не оказывается точно на одной линии с Солнцем, когда центры их дисков совпадают.

Все приведенные выводы были основаны на предположении о сферичности Земли. Для учета ее реальной фигуры, определяемой величиной сжатия, необходимо внести небольшие поправки. Полярный радиус Земли отличается от экваториального радиуса $R_\oplus$ на множитель

$$1 - f = 0{,}996647.$$

Любая точка (x, y, z) поверхности этого эллипсоида вращения удовлетворяет уравнению

$$r^2 = x^2 + y^2 + z^2/(1 - f^2) = R_\oplus^2, \quad (9.8)$$

полученному из (9.5) заменой z на $z/(1 - f)$. Ось z в этой системе координат принимается параллельной оси Земли, то есть мы используем экваториальную систему координат.

Поскольку форма Земли может быть представлена в виде слегка сплющенной сферы, довольно просто вычислить истинные координаты лунной тени. Для этого z-координаты Солнца и Луны умножаются на величину $1/(1 - f)$, через эти координаты проводится прямая и определяется точка ее пересечения со сферой, имеющей радиус Земли. Вычисленная таким образом z-координата больше z-координаты действительной точки пересечения оси тени с земной поверхности также в $1/(1 - f)$ раз.

9.3. Географические координаты и сплюснутость Земли

Географические долгота (λ) и широта (φ) являются наиболее употребительными координатами для описания точки на поверхности Земли. Они тесно связаны

с астрономическими координатами — прямым восхождением (α) и склонением (δ). Обе системы координат связаны с плоскостью экватора и осью вращения Земли.

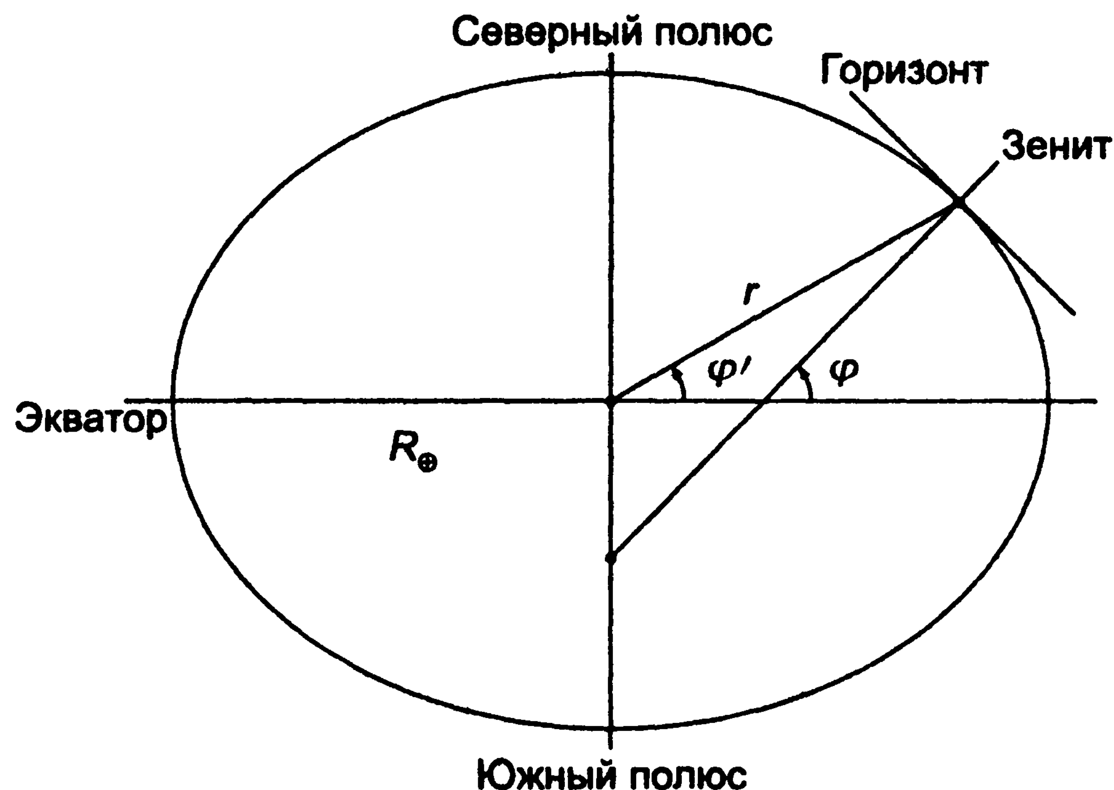


Рис. 9.5. Сплюснутость Земли

Если бы поверхность Земли была идеальной сферой, склонение и географическая широта были бы эквивалентными координатами. Рисунок 9.5 поясняет различие между географической широтой φ, равной углу между земной осью и плоскостью горизонта в некоторой точке, и радиус-вектором, проведенным к этой точке из центра Земли. Поверхность Земли в меридиональном сечении представляет собой эллипс, а плоскость горизонта — касательную к нему. Как видно на рисунке, φ не совпадает с геоцентрической широтой φ' или склонением δ, поскольку обе эти величины представляют собой угол между радиус-вектором и плоскостью экватора. Величины φ и φ' связаны между собой следующими соотношениями

$$\begin{aligned} r \cdot \cos(\varphi') &= \frac{R_\oplus}{\sqrt{1 - e^2 \sin^2 \varphi}} \cos \varphi, \\ r \cdot \sin(\varphi') &= \frac{(1 - e^2) R_\oplus}{\sqrt{1 - e^2 \sin^2 \varphi}} \sin \varphi, \\ \operatorname{tg}(\varphi') &= (1 - e^2) \operatorname{tg} \varphi. \end{aligned} \tag{9.9}$$

Здесь $R_\oplus$ = 6378,14 км — экваториальный радиус Земли, а e — эксцентриситет земного эллипсоида. Центробежная сила вращающейся Земли искажает ее поверхность так, что экваториальный радиус примерно на 20 км больше полярного R_{Pol}. Отношение этих величин определяет сжатие

$$f = \frac{R_\oplus - R_{Pol}}{R_\oplus} = \frac{21{,}385 \text{ км}}{6378{,}14 \text{ км}} = \frac{1}{298{,}257} = 0{,}003353. \tag{9.10}$$

Эксцентриситет земного эллипсоида может быть представлен в виде

$$e = \sqrt{1 - (1 - f)^2} = \sqrt{2f - f^2}. \tag{9.11}$$

Несмотря на точные выражения (9.9), на практике для определения географической широты совершенно достаточно использовать приближение

$$\varphi = \varphi' + 0{,}1924° \cdot \sin(2\varphi'). \qquad (9.12)$$

Из этого следует, что наибольшая разность между φ и φ' существует в средних широтах, где достигает значения около двенадцати угловых минут. На полюсах и экваторе разница исчезает.

Вторая координата точки на земной поверхности — это географическая долгота. По существу, она соответствует прямому восхождению в экваториальной системе координат, но если прямое восхождение отсчитывается от точки весеннего равноденствия — избранного направления в пространстве, то географическая долгота отсчитывается от Гринвичского меридиана. Различие между этими координатами $\Theta_0(t)$ определяется вращением Земли и меняется со временем

$$\alpha = \Theta_0(t) + \lambda. \qquad (9.13)$$

Для позиционных векторов имеем

$$\begin{pmatrix} r\cos(\varphi')\cos(\lambda) \\ r'\cos(\varphi')\sin(\lambda) \\ r\sin(\varphi') \end{pmatrix} = \boldsymbol{R}_z(\Theta_0(t)) \begin{pmatrix} r\cos(\varphi')\cos(\alpha) \\ r\cos(\varphi')\sin(\alpha) \\ r\sin(\varphi') \end{pmatrix}. \qquad (9.14)$$

Величина $\Theta_0(t)$ соответствует прямому восхождению нулевого меридиана и представляет собой попросту звездное время по Гринвичу. Вычисление звездного времени и предназначенная для этих целей процедура GMST уже обсуждались в разделе 3.3.

Существенное затруднение вызывает необходимость определения разницы между Всемирным (UT) и эфемеридным (ET или TDB, TT) временем. Если координаты Солнца и Луны являются функцией равномерного эфемеридного времени и всегда могут быть вычислены, то звездное время получают из Всемирного времени, которое по определению находится из наблюдений, а не по показаниям часов. Поэтому остается лишь пользоваться ранее определенным из наблюдений значением разницы между двумя системами счета времени $\Delta T = \mathrm{ET} - \mathrm{UT}$. Некоторые значения этой величины по измерениям, произведенным в XX веке, представлены в табл. 3.1. Избежать необходимости каждый раз вводить новое ее значение можно, если использовать приближение табличных значений с помощью полиномов.

В табл. 9.2 представлен ряд полиномов, каждый из которых покрывает промежуток в 25 лет, а все вместе они покрывают период с 1825 по 2000 год. Поскольку точность вычислений заметно снижается за пределами указанного времени, функция ETminUT проверяет соответствующие временные границы. Если мы оказываемся за пределами этих границ, переменная valid принимает значение false. Исключение представляет последний полином, позволяющий производить интерполяцию до 2005 года. К сожалению, не представляется возможным

сделать долгосрочное предсказание значения определяемой из наблюдений величины ET–UT. Впрочем, текущие сведения можно найти в различных календарях.

Таблица 9.2. Полиномы для определения разницы ΔT = ET–UT между эфемеридным и Всемирным временем (T = (JD – 2451545)/36525)

Период	ΔT = ET–UT	t
1825–1850	$10\overset{s}{,}4 - 80\overset{s}{,}8t + 413\overset{s}{,}9t^2 - 572\overset{s}{,}3t^3$	$T + 1{,}75$
1850–1875	$6\overset{s}{,}6 + 46\overset{s}{,}4t^2 - 368\overset{s}{,}4t^2 + 18\overset{s}{,}8t^3$	$T + 1{,}50$
1875–1900	$-3\overset{s}{,}9 - 10\overset{s}{,}8t - 166\overset{s}{,}2t^2 + 867\overset{s}{,}4t^3$	$T + 1{,}25$
1900–1925	$-2\overset{s}{,}6 + 114\overset{s}{,}1t + 327\overset{s}{,}5t^2 - 1467\overset{s}{,}4t^3$	$T + 1{,}00$
1925–1950	$24\overset{s}{,}2 - 6\overset{s}{,}3t - 8\overset{s}{,}2t^2 + 483\overset{s}{,}4t^3$	$T + 0{,}75$
1950–1975	$29\overset{s}{,}3 + 32\overset{s}{,}5t - 3\overset{s}{,}8t^2 + 550\overset{s}{,}7t^3$	$T + 0{,}50$
1975–2000	$45\overset{s}{,}3 + 130\overset{s}{,}5t - 570\overset{s}{,}5t^2 + 1516\overset{s}{,}7t^3$	$T + 0{,}25$

```
//------------------------------------------------------------------------------
// ETminUT· разность между эфемеридным и Всемирным временем ET-UT
//   T         время в юлианских столетиях, прошедших после J2000
//   DTsec     ET-UT in [s] - ET-UT в сек
//   valid     Флаг, показывающий, что T находится в области приближения
// Замечание· приближение допустимо только в промежутке с 1825 по 2005 гг.
//------------------------------------------------------------------------------
void ETminUT (double T, double& DTsec, bool& valid)
{
  int    i = (int) floor(T/0.25);
  double t = T-i*0.25;

  if ( (T<-1.75) || (0 05<T) ) {
    valid = false;
    DTsec = 0.0;
  }
  else {
    valid = true;
    switch (i) {
      case -7: DTsec=10.4+t*(-80 8+t*( 413.9+t*( -572.3))); break; // 1825-
      case -6: DTsec= 6.6+t*( 46.3+t*(-358.4+t*(   18.8))); break; // 1850-
      case -5: DTsec=-3.9+t*(-10.8+t*(-166 2+t*(  867 4))); break. // 1875-
      case -4: DTsec=-2.6+t*(114.1+t*( 327.5+t*(-1467.4))); break; // 1900-
      case -3. DTsec=24 2+t*( -6.3+t*(  -8 2+t*(  483 4))); break; // 1925-
      case -2: DTsec=29.3+t*( 32.5+t*(  -3.8+t*(  550.7))); break; // 1950-
      case -1: DTsec=45.3+t*(130.5+t*(-570.5+t*( 1516.7))); break. // 1975-
      case  0: t+=0.25;
               DTsec=45.3+t*(130.5+t*(-570.5+t*( 1516.7)));        // 2000-
    }                                                              // 2005
  }
}
```

Разница между ET и UT изменяется очень медленно, поэтому в течение солнечного затмения ее значение можно считать величиной постоянной.

9.4. Продолжительность затмения

Продолжительность полного или кольцеобразного затмения не одинакова во всех точках центральной линии. Она зависит не только от диаметра лунной тени, но и от скорости, с которой последняя движется по земной поверхности. Для точного определения продолжительности полной фазы необходимо установить с помощью последовательных приближений моменты ее начала и окончания, а такие вычисления довольно громоздки. Ниже излагается метод, позволяющий приблизительно оценить ход затмения и выбрать лучший пункт наблюдений в центральной полосе.

Экваториальные координаты

$$\boldsymbol{r} = (x, y, z) \quad \text{и} \quad \boldsymbol{r}' = (x', y', z')$$

точек пересечения оси лунной тени с земной поверхностью в близкие моменты времени t и $t + \Delta t$ могут быть вычислены с помощью уравнений (9.6) и (9.7)[1]. Разница между этими векторами является мерой скорости движения лунной тени, однако $\boldsymbol{r}' - \boldsymbol{r}$ еще не дает искомое ее перемещение по поверхности Земли.

Период вращения Земли составляет 23^h56^m, поэтому за время Δt Земля поворачивается на угол

$$w = 2\pi \cdot \frac{\Delta t}{1436^m} \quad \text{(радиан).}$$

Точка на земной поверхности, у которой в момент времени $t + \Delta t$ экваториальные координаты $\boldsymbol{r}' = (x', y', z')$, имеет в момент t координаты

$$\boldsymbol{r}'' = \begin{pmatrix} +x' \cdot \cos w + y' \cdot \sin w \\ -x' \cdot \sin w + y' \cdot \cos w \\ +z' \end{pmatrix} \approx \begin{pmatrix} x' + w y' \\ y' - w x' \\ z' \end{pmatrix},$$

которые получаются поворотом $\boldsymbol{r}'$ вокруг оси z на угол w. Таким образом, за время Δt центр лунной тени проходит по земной поверхности путь

$$\Delta \boldsymbol{r} = \boldsymbol{r}'' - \boldsymbol{r}$$

или

$$\begin{pmatrix} \Delta x \\ \Delta y \\ \Delta z \end{pmatrix} = \begin{pmatrix} x' + w y' - x \\ y' - w x' - y \\ z' - z \end{pmatrix}.$$

Пусть $\boldsymbol{e} = (e_x, e_y, e_z)$ — вектор, описывающий направление оси тени, как в выражении (9.4), тогда вектор $\Delta \boldsymbol{r}$ можно разложить на две составляющие: $\Delta \boldsymbol{r}_{\parallel}$, параллельную $\boldsymbol{e}$, и $\Delta \boldsymbol{r}_{\perp}$, перпендикулярную оси тени. Длины этих составляющих равны

$$\Delta r_{\parallel} = \Delta \boldsymbol{r} \cdot \boldsymbol{e} = \Delta x e_x + \Delta y e_y + \Delta z e_z$$

[1] Ввиду некоторых упрощений здесь не учитывается сжатие Земли.

и

$$\Delta r_\perp = \sqrt{(\Delta r)^2 - (\Delta r_\parallel)^2}\,.$$

$\Delta r_\perp$ равняется длине пути точки лунной тени в направлении, перпендикулярном ее оси за время Δt. Продолжительность полной или кольцевой фазы при диаметре тени $|d|$ составляет

$$\tau = \frac{|d|}{r_\perp} \cdot \Delta t\,.$$

9.5. Координаты Солнца и Луны

Знание точных координат Солнца и Луны необходимо для предвычисления солнечных затмений. Чтобы определить положение линии центрального затмения с точностью, например, 10 км, необходимо знать положение Луны на орбите с такой же примерно точностью. При расстоянии в 380 000 км это соответствует углу в 5″. Такие же требования предъявляются и к геоцентрическим координатам Солнца. Такую точность обеспечивают обсуждавшиеся ранее процедуры SunPos и MoonPos. Здесь нет необходимости вновь возвращаться к рассмотренным в главах 6 и 8 рядам, описывающим возмущения орбит.

Для наших целей желательно использовать процедуру, обеспечивающую не только требуемую точность, но и достаточную скорость, поскольку при вычислении хода затмения требуется много раз определять координаты Солнца и Луны. Поэтому сначала получим разложение Чебышева для индивидуальных координат. Под этим мы подразумеваем получение соответствующих полиномов, описывающих орбиты Солнца и Луны с достаточной точностью в небольшие промежутки времени. Сделать такую оценку значительно проще, нежели завершить общепринятое разложение в ряд. Этот метод использовался при вычислении лунных эфемерид (глава 6). Основы разложений Чебышева и соответствующий класс Cheb3D обсуждались там же, и читатель должен быть вполне знаком с ними.

Чтобы получить разложение для координат Луны, требуется лишь задать объект класса Cheb3D. В конструкторе должны быть указаны функция, порядок разложения и интервал, в котором осуществляется приближение

```
// Константы
const double Interval - 1.0/36525 0;   // 1d (в столетиях)

// Глобальная дата лунных эфемерид
Cheb3D ChebMoonEqu(MoonEqu.10.Interval).
```

Поскольку продолжительность солнечных затмений не превышает нескольких часов, достаточно при разложении ограничиться полиномами десятого порядка, покрывающими промежуток в одни сутки. Более того, очевидно, что функция MoonEqu может быть использована вместо MoonPos, поскольку вычисление солнечных затмений производится в экваториальной системе координат. Этот метод не требует дополнительного перевода координат из эклиптической систе-

мы в экваториальную и обратно. Полиномы для координат Луны могут быть вычислены для любого момента времени с помощью метода ChebMoonEqu.Value(T):

```
r_Moon = ChebMoonEqu.Value(T_ET);   // км
```

Следует отметить, что вычисление приближения Чебышева автоматически производится при первом обращении к Value, если только пользователь не обратился прежде к методу Fit.

Аналогично для получения координат Солнца мы используем другую небольшую функцию SunEqu, вычисляющую истинные экваториальные координаты Солнца в заданный момент времени. В этом случае нутация (см. раздел 6.4.2) учитывается после преобразования эклиптических координат Солнца в экваториальные. Полученный таким образом позиционный вектор представляет координаты Солнца в системе координат, связанной с реальным, а не осредненным положением экватора и оси Земли. Функция MoonEqu так же учитывает нутацию при вычислении лунной орбиты.

```
//------------------------------------------------------------------------
// SunEqu: вычисление экваториальных координат Солнца аналитическими методами
//   T - время в юлианских столетиях, прошедших после J2000
// Возвращаемые значения: геоцентрические координаты Солнца, отнесенные
// к положению экватора и равноденствию даты
//------------------------------------------------------------------------
Vec3D SunEqu ( double T )
{
  return NutMatrix(T) * Ecl2EquMatrix(T) * SunPos(T),
}
```

Разложение Чебышева для координат Солнца вновь производится объявлением соответствующего объекта Cheb3D:

```
Cheb3D ChebSunEqu(SunEqu,5,Interval),
```

Медленное движение Солнца позволяет ограничиться первыми членами разложения.

При сравнении положений Солнца и Луны должно учитываться еще одно обстоятельство. Время распространения света от Солнца до Земли составляет 8,32 минуты, за это время Солнце смещается на 20″. Так как *видимое* положение Солнца в момент t соответствует его геометрическому положению в момент испускания света $t - 8^{m}_{,}32$, при вычислении координат функция SunEqu (как и ChebSunEqu.Value) вносит соответствующую поправку.

```
const double tau_Sun = 8 32 / ( 1440 0*36525.0);   // 8 32 мин (в столетиях)
r_Sun = ChebSunEqu.Value (T_ET-tau_Sun)*AU;         // км
```

В случае Луны этот эффект учитывается в теории Брауна, поэтому нет необходимости вносить какие-либо поправки.

9.6. Программа Eclipse

Программа Eclipse по дате новолуния рассчитывает полосу полной фазы и продолжительность полного или кольцевого затмения.

Ядром программы является функция Intersect, вычисляющая по геоцентрическим координатам Солнца и Луны точки пересечения оси лунной тени с земной поверхностью. Также вычисляются диаметры тени и полутени, из которых определяются соответствующие фазы затмения. Функция Central преобразует экваториальные координаты тени, вычисленные Intrinsect, в географические и определяет продолжительность полной фазы.

Программа запрашивает дату новолуния, разность между Всемирным и эфемеридным временем и размер шага при выводе данных. Величина ΔT = ET − UT известна только для предшествующего времени и на будущее может лишь оцениваться. Обычно при вычислении затмений полагается ET − UT = 0. Вычисленные координаты точек линии центрального затмения смещены по долготе относительно их истинных значений — разница соответствует повороту Земли за время ΔT. Если λ' — вычисленная долгота (возрастающая к востоку), истинная долгота равна:

$$\lambda = \lambda' + 0{,}^\circ25/^{m}\cdot\Delta T.$$

В настоящее время величина $\Delta T \approx 1^m$, и координаты, вычисленные в предположении $\Delta T = 0$, соответствуют точкам, лежащим примерно на 1/4° восточнее. На экваторе это соответствует примерно 28 км.

```
//------------------------------------------------------------------------
// Модуль· программа Eclipse (Eclipse.cpp)
// Задача модуля: вычисление положения линии центрального затмения
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//------------------------------------------------------------------------

#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Cheb.h"
#include "APC_Const.h"
#include "APC_IO.h"
#include "APC_Math.h"
#include "APC_Moon.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

using namespace std:

// Определение типов данных
enum enPhase { NoEclipse. partial. NonCenAnn. NonCenTot. annular. total }.
// Константы
const double Interval = 1.0/36525.0:    // 1d [cy]
// Общие эфемеридные данные для Луны и Солнца
Cheb3D ChebMoonEqu(MoonEqu.10.Interval):
Cheb3D ChebSunEqu (SunEqu.  5.Interval):
```

```
//---------------------------------------------------------------------------
// Вычисление координат точки пересечения оси тени с земной поверхностью, определение
единичного вектора оси, диаметра тени и фазы
//   T_ET - эфемеридное время в юлианских столетиях, прошедших после J2000
//   r - точка пересечения оси тени с земной поверхностью
//   (геоцентрические координаты, км)
//   e - единичный вектор оси
//   D_umbra - диаметр тени в главной плоскости
//   Phase - фаза затмения
//---------------------------------------------------------------------------
void Intersect ( double T_ET,
                 Vec3D& r, Vec3D& e, double& D_umbra, enPhase& Phase )
{
  const double tau_Sun  = 8.32 / ( 1440.0*36525.0);    // 8.32 min  [cy]
  const double fac      = 0.996647; // Отношение полярного и экваториального радиусов Земли
double r_MS,s0,Delta,r0,s;
  double D_penumbra;
  Vec3D  r_Moon;
  Vec3D  r_Sun;
  // координаты Солнца и Луны относительно истинного экватора и равноденствия даты
  // (с учетом времени распространения света)
  r_Moon = ChebMoonEqu.Value(T_ET);               // [km]
  r_Sun  = ChebSunEqu.Value (T_ET-tau_Sun)*AU;  // [km]

  // поправка к z-координате за сжатие Земли
  r_Moon = Vec3D ( r_Moon[x], r_Moon[y], r_Moon[z]/fac );
  r_Sun  = Vec3D ( r_Sun[x] , r_Sun[y] , r_Sun[z]/fac  );

  // ось тени (единичный вектор Солнце-Луна
  r_MS = Norm(r_Moon-r_Sun);
  e    = (r_Moon-r_Sun)/r_MS;

  // расстояние от главной плоскости до Луны
  s0 = -Dot(r_Moon,e);

  // расстояние от центра Земли до оси тени
  Delta = s0*s0 + R_Earth*R_Earth - Dot(r_Moon,r_Moon);
  r0 = sqrt(R_Earth*R_Earth-Delta);

  // диаметры тени и полутени в главной плоскости
  D_umbra    = 2.0*((R_Sun-R_Moon)*(s0/r_MS)-R_Moon);
  D_penumbra = 2.0*((R_Sun-R_Moon)*(s0/r_MS)+R_Moon);

  // определение фазы и координат тени в случае необходимости
  if (r0<R_Earth) {

    // ось тени пересекает земную поверхность: полное или кольцевое затмение

    // пересечение оси тени с земной поверхностью
    s = s0-sqrt(Delta);
    r = r_Moon + s*e;

    // преобразованная z-координата
    r = Vec3D(r[x],r[y],fac*r[z]);

    // диаметр тени на поверхности Земли
```

```
    D_umbra = 2.0*((R_Sun-R_Moon)*(s/r_MS)-R_Moon);

    Phase = ( (D_umbra>0.0)? annular : total );
  }
  else {

    if ( r0 < R_Earth+0.5*fabs(D_umbra) ) {
      // нецентральное затмение
      Phase = ( (D_umbra>0.0)? NonCenAnn : NonCenTot );
    }
    else {
      // частное затмение или затмение невозможно
      Phase = ( (r0<R_Earth+0.5*fabs(D_penumbra))? partial : NoEclipse );
    };

    r = Vec3D(); // Null vector
  };

};

//------------------------------------------------------------------------------
//
// Вычисление координат линии центрального затмения, его продолжительности и фазы
// MjdUT Всемирное время в виде модифицированной юлианской даты
// ET_UT Разность между эфемеридным и Всемирным временем, с
// Lambda Географическая долгота центра тени (направление на восток положительно), рад
// Phi Географическая широта центра тени, рад
// Durat Продолжительность полной фазы в пункте с координатами лямбда и фи
// Phase Фаза затмения
//------------------------------------------------------------------------------
void Central ( double MjdUT, double ET_UT,
               double &Lambda, double& Phi, double& Durat, enPhase& Phase )
{
  const double dt    = 0.1  / (1440.0*36525 0), // 0.1 min [cy]
  const double omega = pi2 * 1 002738*36525.0;  // [rad/cy]

  double    T_ET;             // эфемеридное время
  Vec3D     r,r_G,rr,dr;      // координаты тени
  Vec3D     e,ee;             // ось тени
  double    D_umbra, DU;      // диаметр тени
  enPhase   Ph=NoEclipse;     // фаза
  double    w;                // угол поворота Земли
  double    drp;              // смещение тени

  // эфемеридное время
  T_ET = ( MjdUT + ET_UT/86400.0 - MJD_J2000 ) / 36525 0,

  // пересечение оси тени с земной поверхностью
  Intersect ( T_ET, r, e, D_umbra, Phase );

  // только для центрального затмения: географические координаты и продолжительность
  // полной фазы
  Lambda = Phi = Durat = 0.0,

  if ( Phase >= annular ) {

    // географические координаты тени
```

```
    r_G    = R_z(GMST(MjdUT))*r;              // координаты Гринвича
    Lambda = Modulo(r_G[phi]+pi,2*pi)-pi;     // восточная долгота
    Phi    = r_G[theta];                      // геоцентрическая широта
    Phi    = Phi + 0.1924*Rad*sin(2*Phi),     // географическая широта

    // продолжительность полной фазы для данного пункта

    // координаты тени в момент времени T+dt (или T-dt)
    if (Ph<annular) {
      Intersect ( T_ET-dt, rr, ee, DU, Ph ); w = -dt*omega,
    }
    else {
      Intersect ( T_ET+dt, rr, ee, DU, Ph ); w = +dt*omega;
    }

    // смещение тени dr по поверхности Земли и его составляющая drp,
    // перпендикулярная оси тени
    dr  = Vec3D ( rr[x]-r[x]+w*r[y], rr[y]-r[y]-w*r[x], rr[z]-r[z] );
    drp = Norm ( dr - Dot(dr,e)*e );

    Durat = (36525.0*1440.0)*dt * fabs(D_umbra) / drp,  // [min]
  };

},

//------------------------------------------------------------------------------
// GetInput: запрос пользователю о приблизительной дате и времени новолуния,
// величине шага и разности ET-UT
//   MjdStart  начало промежутка времени для вычисления затмения
//   Step      размер шага, сут.
//   MjdEnd    конец промежутка времени для вычисления затмения
//   ET_UT     разность между эфемеридным и Всемирным временем, с
//------------------------------------------------------------------------------
void GetInput(double& MjdStart, double& Step, double& MjdEnd, double& ET_UT)
{
  int    year, month, day, step;
  double hour, MJD;
  bool   valid;

  // дата и величина шага
  cout << endl
       << " Date of New Moon (yyyy mm dd hh.h)      . . ";
  cin >> year >> month >> day >> hour; cin.ignore(81,'\n'),

  cout << " Output step size (min)" << setw(21) << right << " .. ";
  cin >> step, cin.ignore(81,'\n');

  // Дата, округленная до величины шага
  Step = step/1440.0;  // [d]

  MJD = Mjd(year,month,day) + floor((hour/24.0)/Step+0 5)*Step;

  // начальная и конечная даты (дата новолуния +/- 6 часов)
  MjdStart = MJD - 0.25;
  MjdEnd   = MJD + 0.25;

  // разность между эфемеридным и Всемирным временем
```

```
  ETminUT ( (MJD-MJD_J2000)/36525.0, ET_UT, valid );
  if ( valid ) {
    cout << " Difference ET-UT (proposal:" << fixed << setw(6)
         << setprecision (1) << ET_UT << " sec) ... ";
    cin >> ET_UT; cin.ignore(81,'\n');
  }
  else {
    cout << " Difference ET-UT (sec)" << setw(21) << right << "... ";
    cin >> ET_UT; cin.ignore(81,'\n');
  }
}

//------------------------------------------------------------------------------
//
// Main program - основная программа
//
//------------------------------------------------------------------------------
void main(int argc, char* argv[]) {

// Variables - переменные
  double    MjdStart, Step, MjdEnd;
  double    MjdUT, ET_UT;
  double    Lambda, Phi, Durat;
  enPhase   Phase;
  char      OutputFile[APC_MaxFilename] = "";
  ofstream  OutFile;

  // Заставка
  cout << endl
       << "    ECLIPSE: central line and duration of solar eclipses " << endl
       << "          (c) 1999 Oliver Montenbruck, Thomas Pfleger     " << endl
       << endl;

  // Приглашение к вводу данных
  GetInput ( MjdStart, Step, MjdEnd, ET_UT );

  // выбор файла ввода и переадресация ввода
  if (argc==2) {
    strncpy(OutputFile, argv[1], APC_MaxFilename);
    OutFile.open(OutputFile);
    if (OutFile.is_open())
      cout = OutFile;
  }

  // Заголовок
  cout << endl << endl
       << "    Date       UT          Phi      Lambda    Durat  Phase " << endl
       << "               h  m         o  '      o  '     min         " << endl
       << endl;

  // Фаза и линия центрального затмения
  MjdUT = MjdStart;

  // Начало цикла
  while ( MjdUT < MjdEnd + Step/2 ) {

    // Фаза и положение тени
```

```
    Central ( MjdUT. ET_UT. Lambda. Phi. Durat. Phase ):

    // Вывод данных
    if ( Phase != NoEclipse ) {

      cout << " " << DateTime(MjdUT.HHMM):

      if ( Phase < annular )
        cout << "        -- --     -- --    ---".
      else {
        cout << showpos
             << "      " << setw(6) << Angle(Deg*Phi.DMM)
             << "   "    << setw(7) << Angle(Deg*Lambda.DMM)
             << noshowpos
             << fixed << setprecision(1) << setw(7) << Durat.
      }.

      switch(Phase) {

        case partial  : cout << "  partial              ". break:
        case NonCenAnn: cout << "  annular (non-central) ": break.
        case NonCenTot: cout << "  total (non-central)   ": break:
        case annular  . cout << "  annular              ". break:
        case total    : cout << "  total                ":
      }:

      cout << endl:
    }:

    MjdUT += Step:  // Next time step

  }:  // End time loop

  if (OutFile.is_open()) OutFile close():
}
```

Для примера рассчитаем путь лунной тени во время полного затмения, которое будет наблюдаться на территории Соединенных Штатов Америки в 2017 году. Полоса полной фазы этого затмения проходит по штатам Орегон, Айдахо, Вайоминг, Небраска, Миссури, Теннеси и Каролина. Вначале с помощью Phases определим даты новолуний в 2017 году и выберем те, в которые затмение возможно.

```
          PHASES: Phases of the Moon and Check for Eclipses
             (c) 1999 Oliver Montenbruck. Thomas Pfleger


 Dates of Lunar Phases for the year  ..  2017

 New Moon           First Quarter      Full Moon          Last Quarter
 2016/12/29 06:54   2017/01/05 19:48   2017/01/12 11:35   2017/01/19 22·15
 2017/01/28 00:08   2017/02/04 04.20   2017/02/11 00.34p? 2017/02/18 19·34
 2017/02/26 15:00c  2017/03/05 11:33   2017/03/12 14:55   2017/03/20 15:59
 2017/03/28 02:58   2017/04/03 18:41   2017/04/11 06:09   2017/04/19 09.58
 2017/04/26 12:17   2017/05/03 02:48   2017/05/10 21:44   2017/05/19 00:34
 2017/05/25 19.46   2017/06/01 12:43   2017/06/09 13:11   2017/06/17 11.34
 2017/06/24 02:32   2017/07/01 00:52   2017/07/09 04:08   2017/07/16 19:27
```

```
2017/07/23 09.47    2017/07/30 15·24    2017/08/07 18.12p  2017/08/15 01·16
2017/08/21 18:31c   2017/08/29 08.14    2017/09/06 07·04   2017/09/13 06.26
2017/09/20 05:31    2017/09/28 02.55    2017/10/05 18 41   2017/10/12 12·27
2017/10/19 19·13    2017/10/27 22:23    2017/11/04 05:24   2017/11/10 20 38
2017/11/18 11:43    2017/11/26 17·04    2017/12/03 15·48   2017/12/10 07.52
2017/12/18 06 32    2017/12/26 09·21    2018/01/02 02·25   2018/01/08 22.26
2018/01/17 02.18    2018/01/24 22:21    2018/01/31 13·28t  2018/02/07 15.55

All times in Ephemeris Time (ET)
```

Метка «с» против дат 26 февраля и 21 августа означает, что эклиптическая широта Луны достаточно мала, чтобы могло произойти центральное затмение (ср. табл. 9.1). Более тщательное исследование с помощью программы Eclipse показывает, что затмения наблюдаются в следующих регионах:

26 февраля 2017: южная часть Тихого океана, юг Чили и Аргентины, южная Атлантика, Ангола (затмение кольцевое).

21 августа 2017: северная часть Тихого океана, центральные районы США, северная Атлантика (затмение полное).

Путь тени во время полного затмения 21 августа 2017 также может быть определен с помощью Eclipse. Для этого введем полученную ранее дату новолуния, размер шага вывода данных в минутах и разницу между эфемеридным и Всемирным временем. Для вычисляемого затмения примем последнюю величину равной 80 секундам. В нижеследующем примере все данные, введенные пользователем, выделены курсивом. Чтобы не приводить несоразмерно длинную распечатку данных, сведения для частных фаз приводятся в сокращенном варианте, а размер шага изменен до трех минут.

```
   ECLIPSE: central line and duration of solar eclipses
         (c) 1999 Oliver Montenbruck, Thomas Pfleger


Date of New Moon (yyyy mm dd hh h)      .  2017 08 21 18 5
Output step size (min)                  .  1
Difference ET-UT (sec)                   . 80

   Date        UT         Phi     Lambda   Durat  Phase
              h  m        o  '     o  '    min
2017/08/21 15:47        -- --    -- --    ---   partial
2017/08/21 15.48        -- --    -- --    ---   partial
   .  .  .  .  .          .        .        .
2017/08/21 16.48        -- --    -- --    ---   partial
2017/08/21 16 49       +40 21  -168 41    0.0   total
2017/08/21 16·51       +42 18  -158 14    1.2   total
2017/08/21 16:54       +43 24  -150 52    1.3   total
2017/08/21 16:57       +44 02  -145 33    1.5   total
2017/08/21 17·00       +44 26  -141 11    1.6   total
2017/08/21 17:03       +44 42  -137 24    1.7   total
2017/08/21 17:06       +44 51  -134 01    1 8   total
2017/08/21 17:09       +44 56  -130 56    1 8   total
2017/08/21 17:12       +44 56  -128 06    1.9   total
2017/08/21 17.15       +44 54  -125 28    2.0   total
2017/08/21 17:18       +44 49  -123 00    2 1   total
2017/08/21 17:21       +44 41  -120 40    2 1   total
```

```
2017/08/21 17:24      +44 32   -118 28    2.2   total
2017/08/21 17:27      +44 20   -116 22    2.2   total
2017/08/21 17:30      +44 07   -114 23    2.3   total
2017/08/21 17:33      +43 53   -112 28    2.4   total
2017/08/21 17:36      +43 36   -110 38    2.4   total
2017/08/21 17:39      +43 19   -108 53    2.4   total
2017/08/21 17:42      +43 01   -107 11    2.5   total
2017/08/21 17:45      +42 41   -105 34    2.5   total
2017/08/21 17:48      +42 21   -103 59    2.6   total
2017/08/21 17:51      +41 59   -102 28    2.6   total
2017/08/21 17:54      +41 37   -100 59    2.6   total
2017/08/21 17:57      +41 13   - 99 33    2.6   total
2017/08/21 18:00      +40 49   - 98 10    2.7   total
2017/08/21 18:03      +40 24   - 96 48    2.7   total
2017/08/21 18:06      +39 59   - 95 29    2.7   total
2017/08/21 18:09      +39 33   - 94 12    2.7   total
2017/08/21 18:12      +39 06   - 92 56    2.7   total
2017/08/21 18:15      +38 38   - 91 42    2.7   total
2017/08/21 18.18      +38 10   - 90 29    2.7   total
2017/08/21 18:21      +37 41   - 89 18    2.7   total
2017/08/21 18:24      +37 12   - 88 08    2.7   total
2017/08/21 18:27      +36 42   - 86 59    2.7   total
2017/08/21 18:30      +36 11   - 85 51    2.7   total
2017/08/21 18.33      +35 40   - 84 43    2.7   total
2017/08/21 18:36      +35 08   - 83 36    2 7   total
2017/08/21 18:39      +34 36   - 82 30    2 7   total
2017/08/21 18:42      +34 03   - 81 24    2.7   total
2017/08/21 18:45      +33 30   - 80 19    2.7   total
2017/08/21 18:48      +32 56   - 79 14    2.6   total
2017/08/21 18:51      +32 22   - 78 08    2.6   total
2017/08/21 18:54      +31 46   - 77 03    2.6   total
2017/08/21 18:57      +31 11   - 75 57    2.5   total
2017/08/21 19:00      +30 34   - 74 50    2.5   total
2017/08/21 19:03      +29 57   - 73 43    2.5   total
2017/08/21 19:06      +29 20   - 72 36    2.4   total
2017/08/21 19:09      +28 41   - 71 27    2.4   total
2017/08/21 19:12      +28 02   - 70 17    2.3   total
2017/08/21 19:15      +27 22   - 69 05    2.3   total
2017/08/21 19:18      +26 41   - 67 51    2.3   total
2017/08/21 19:21      +25 59   - 66 36    2.2   total
2017/08/21 19:24      +25 17   - 65 17    2.1   total
2017/08/21 19:27      +24 33   - 63 56    2.1   total
2017/08/21 19:30      +23 48   - 62 31    2.0   total
2017/08/21 19·33      +23 01   - 61 01    2.0   total
2017/08/21 19·36      +22 13   - 59 27    1.9   total
2017/08/21 19:39      +21 23   - 57 46    1.8   total
2017/08/21 19:42      +20 31   - 55 56    1.7   total
2017/08/21 19:45      +19 37   - 53 57    1.7   total
2017/08/21 19:48      +18 39   - 51 45    1 6   total
2017/08/21 19·51      +17 36   - 49 14    1.5   total
2017/08/21 19.54      +16 27   - 46 17    1 4   total
2017/08/21 19·57      +15 08   - 42 33    1.2   total
2017/08/21 20:00      +13 27   - 37 07    1.1   total
2017/08/21 20:02       -- --     -- --    ---   total (non-central)
2017/08/21 20:03       -- --     -- --    ---   partial
  .   .   .   .   .       .         .      .    .
2017/08/21 21:03       -- --     -- --    ---   partial
2017/08/21 21:04       -- --     -- --    ---   partial
```

Если затмение полное, но не центральное, ось лунной тени проходит мимо Земли. Однако в некоторых пунктах конус тени все же касается земной поверхности. Обычно это происходит в начале или в конце затмения. Нецентральные полные или кольцевые затмения очень редки и неудобны для наблюдений как из-за малой продолжительности, так и из-за того, что они наблюдаются в высоких географических широтах.

9.7. Обстоятельства затмений

Сведения об обстоятельствах затмения в конкретном пункте, не менее важные, чем сведения о положении линии центрального затмения, имеют существенное значение для организации наблюдений.

Помимо наибольшей фазы весьма важно знать ее продолжительность и величину, а также моменты контактов, когда диски Солнца и Луны соприкасаются. Первый и четвертый контакты соответствуют моментам начала и окончания частных фаз, когда полутень накрывает или покидает пункт наблюдения. Они определяют полную продолжительность затмения. Второй и третий контакты, известные еще как внутренние контакты, определяют продолжительность полной или кольцевой фазы.

Сведения о контактах обычно дополняются информацией о значении позиционных углов. В случае первого и четвертого контактов позиционный угол, измеряемый от направления на север против часовой стрелки, указывает точку на солнечном лимбе, в которой Луна впервые касается его или, напротив, покидает солнечный диск. Позиционный угол внутренних контактов указывает точки, в которых при полном затмении солнечный диск исчезает или вновь открывается, а в случае кольцевого затмения указывает точки, в которых вокруг Луны образуется или разрывается замкнутое кольцо.

Первый и четвертый контакты наблюдаются на всем протяжении искривленной границы лунной полутени, второй и третий — лишь если наблюдатель находится на границе тени.

Как при последующем рассмотрении обстоятельств покрытия звезд Луной (см. раздел 10.3), в нашем случае будет удобнее пользоваться системой координат, связанной с главной плоскостью, перпендикулярной оси лунной тени и проходящей через центр Земли (см. рис. 9.4). Оси x и y лежат в этой плоскости, а ось z параллельна прямой, соединяющей центры Луны и Солнца. Ось x, кроме того, проходит через точку пересечения земного экватора с главной плоскостью. Единичные векторы $\boldsymbol{i}$, $\boldsymbol{j}$ и $\boldsymbol{k}$, ориентированные вдоль осей x, y и z, могут быть представлены в виде

$$\boldsymbol{i} = \begin{pmatrix} -k_y/\sqrt{k_x^2 + k_y^2} \\ +k_x/\sqrt{k_x^2 + k_y^2} \\ 0 \end{pmatrix}, \quad \boldsymbol{j} = \boldsymbol{k} \times \boldsymbol{i}, \quad \boldsymbol{k} = \frac{\boldsymbol{r}_\odot - \boldsymbol{r}_M}{|\boldsymbol{r}_\odot - \boldsymbol{r}_M|}, \qquad (9.15)$$

где векторы $r_\odot$ и r_M представляют экваториальные координаты Солнца и Луны. Проекция r_M на базисные векторы $\boldsymbol{i}, \boldsymbol{j}$ и $\boldsymbol{k}$ дает следующие координаты Луны в главной плоскости

$$\begin{pmatrix} x_M \\ y_M \\ z_M \end{pmatrix} = \begin{pmatrix} \boldsymbol{r}_M \cdot \boldsymbol{i} \\ \boldsymbol{r}_M \cdot \boldsymbol{j} \\ \boldsymbol{r}_M \cdot \boldsymbol{k} \end{pmatrix}.$$

Поскольку по определению линия, соединяющая Солнце и Луну, параллельна оси z, ось лунной тени в главной плоскости имеет координаты x_M и y_M. Сечения тени и полутени главной плоскостью представляют собой круги, радиусы которых в соответствии с (9.3) равны:

$$l = z_M \frac{R_\odot \mp R_M}{r_{\odot M}} \mp R_M. \tag{9.16}$$

Здесь $R_\odot$ и R_M — радиусы Солнца и Луны, а $r_{\odot M} = |r_\odot - r_M|$ — расстояние между ними. Минус относится к тени, плюс — к полутени.

Чтобы определить момент, когда наблюдатель оказывается на границе той или другой области тени, его геоцентрические координаты проектируются на главную плоскость. Как было показано в разделе 9.3, экваториальные координаты наблюдателя, находящегося в точке с геоцентрической долготой λ, геоцентрической широтой φ' и на расстоянии r от центра Земли, равны:

$$\boldsymbol{r}_0 = \boldsymbol{R}_z(-\Theta_0(t)) \begin{pmatrix} r\cos(\varphi')\cos(\lambda) \\ r\cos(\varphi')\sin(\lambda) \\ r\sin(\varphi') \end{pmatrix}, \tag{9.17}$$

где $\Theta_0(t)$ — звездное время по Гринвичу. Координаты наблюдателя (x_O, y_O, z_O) в системе, связанной с главной плоскостью, находятся путем проекции $\boldsymbol{r}_O$ на базисные векторы $\boldsymbol{i}, \boldsymbol{j}$ и $\boldsymbol{k}$. Если расстояние между наблюдателем и главной плоскостью равно z_O, радиус лунной тени в угловой мере равен f, а ее линейный диаметр в главной плоскости равен l, можно вычислить радиус тени в пункте наблюдения:

$$L = l - z_O \operatorname{tg} f. \tag{9.18}$$

В случае частного или кольцевого затмения L всегда положительно, тогда как радиус области тени при полном затмении всегда отрицателен.

Наблюдатель находится на поверхности конуса тени, если расстояние между ним и ее осью равно радиусу тени в пункте наблюдения. Это происходит в тот момент, когда расстояние, определяемое функцией

$$f(t) = (x_M - x_O)^2 + (y_M + y_O)^2 - L^2, \tag{9.19}$$

равно нулю. Моменты контактов, таким образом, определяются нахождением точек, в которых $f(t)$ обращается в нуль. Момент наибольшей фазы может быть определен нахождением минимума той же функции $f(t)$.

Поскольку взаимное расположение наблюдателя и оси тени непосредственно отражает видимое с позиций наблюдателя расположение Солнца и Луны, позиционные углы контактов могут быть просто определены по координатам в главной плоскости. Позиционный угол отсчитывается от направления на север против часовой стрелки, поэтому, независимо от типа затмения, позиционные углы внешних и внутренних контактов находят из следующих соотношений:

$$\begin{pmatrix} \cos P \\ \sin P \end{pmatrix} = \begin{pmatrix} (y_M - y_O)/L \\ (x_M - x_O)/L \end{pmatrix}. \quad (9.20)$$

Решая эти уравнения, важно помнить, что, как уже было сказано, радиус тени при полном затмении имеет отрицательное значение.

Кроме позиционного угла P, отсчитываемого от направления на север, полезно также знать позиционный угол контактов относительно направления в зенит, $V = P - C$. Угол C соответствует позиционному углу точки зенита и может быть получен с достаточной точностью из координат наблюдателя в главной плоскости:

$$\begin{pmatrix} \cos C \\ \sin C \end{pmatrix} \approx \begin{pmatrix} y_O/\sqrt{x_O^2 + y_O^2} \\ x_O/\sqrt{x_O^2 + y_O^2} \end{pmatrix}. \quad (9.21)$$

Продолжительность затмения можно узнать также по значению наибольшей фазы[1] M, которая измеряется в долях солнечного диска, заслоненного Луной. Величина M может быть получена непосредственно из значений радиусов L_1 и L_2 полутени и тени в пункте наблюдения. Для полного и кольцевого затмений

$$M_2 = \frac{L_1 - L_2}{L_1 + L_2},$$

что представляет собой отношение видимых радиусов Солнца и Луны. Для частного затмения имеем

$$M_1 = \frac{L_1 - m}{L_1 + L_2}.$$

В этом случае наибольшая фаза определяется также расстоянием m между наблюдателем и осью тени. Следует заметить, что величина M не тождественна доле покрытого Луной солнечного диска и, в частности, при полном затмении принимает значение больше единицы.

9.8. Программа Ecltimer

Хотя положение линии центрального затмения обычно известно и сведения об этом заблаговременно публикуются, календари не содержат подробной информа-

[1] В отечественной литературе для обозначения наибольшей фазы используется значок Φ_m. — *Примеч. перев.*

ции об обстоятельствах затмения в конкретных пунктах. Мы опишем программу EclTimer, представляющую собой дополнение к программе Eclipse и позволяющую рассчитать обстоятельства затмения для любого избранного пункта.

Ради экономии места мы опишем лишь основную структуру программы, использующей методы, изложенные в предыдущем разделе. Прилагаемый к книге компакт-диск содержит полный исходный код с подробными комментариями[1].

Моменты контактов определяет функция Contacts, вычисляющая точки, в которых обращается в нуль функция $f(t) = m^2(t) - L^2(t)$, где m — расстояние между наблюдателем и осью тени, а L — радиус тени. Расстояние до тени вычисляет функция ShadowDist, включающая в себя две процедуры Bessel и Observer, определяющие положение главной плоскости, параметры конуса тени и координаты наблюдателя.

Далее путем квадратичной интерполяции находятся моменты первого и четвертого контактов, когда наблюдатель оказывается на поверхности конуса тени. Для этого используется процедура Quad, обсуждавшаяся в главе 3, она же приближает нас к вычислению момента, когда расстояние между наблюдателем и осью тени становится наименьшим. Таким образом, нам становятся известны обстоятельства частного затмения, далее функция Contacts определяет наибольшую фазу.

Если расстояние между наблюдателем и осью тени, а также диаметр последней позволяют увидеть полное или кольцевое затмение, можно определить моменты второго и третьего контактов. Поскольку длительность полного или кольцевого затмения никогда не превышает тринадцати минут, моменты второго и третьего контактов находятся в пределах соответствующего промежутка времени вблизи наибольшей фазы. Это позволяет использовать процедуру, аналогичную методу regula falsi, позволяющую быстрее достичь результата, нежели методом квадратичной интерполяции. В программе EclTimer мы используем вариант *regula falsi*, известный как метод Pegasus (см. раздел 10.2).

Наконец, процедура PosAngles определяет позиционные углы внешних и внутренних контактов как относительно направления на север, так и относительно направления в зенит.

При запуске программа EclTimer сперва запрашивает дату новолуния, значение разницы между ET и UT, ΔT = ET − UT. Затем вводятся географические координаты пункта наблюдения. С помощью программы Phases можно получить сведения о времени наступления затмений.

В качестве примера вычислим обстоятельства кольцевого затмения 10 мая 1994 года. Выберем в качестве пункта наблюдений город Рабат в Марокко, имеющий координаты $\lambda = 6^\circ\!.8333$ западной долготы и $\varphi = 33^\circ\!.95$. Предположим, что для этой даты разность ΔT равнялась 60^s. Вводимые данные как обычно выделены курсивом.

[1] Согласно лицензионному соглашению все тексты программ на компакт-диске приводятся без изменений, то есть с комментариями на английском языке. — *Примеч. ред.*

```
     ECLTIMER: local circumstances of solar eclipses
       (c) 1999 Oliver Montenbruck, Thomas Pfleger

 Date of new Moon (yyyy mm dd hh.hhh)         ... 1994 05 10 17.1
 Difference ET-UT (proposal:  60.2 sec)       ... 60.0
 Observer's coordinates: East longitude [deg] ... -6.8333
                              latitude [deg]  ... +33.95

 Annular eclipse with M=0.931 (0.87)
 Maximum at 1994/05/10 18:58:42 UT

                                        h  m  s [UT]    P [o]     V [o]
 1st contact (1 контакт): 1994/05/10 17:50:47            267       207
 2nd contact (2 контакт): 1994/05/10 18:56:40            235       179
 3rd contact (3 контакт): 1994/05/10 19:00:29            120        65
 4th contact (4 контакт): 1994/05/10 19:59:19             89        38
```

В течение кольцевой фазы, длящейся в выбранном нами пункте около четырех минут, видимый диаметр Луны на 7 % меньше солнечного. Таким образом, Луна закрывает лишь 87 % видимой поверхности Солнца. Поскольку затмение происходит вечером, позиционный угол относительно направления в зенит (V) примерно на 50–60° меньше позиционного угла относительно направления на север (P). Следует отметить, что в этот день в Рабате Солнце садится в 19^h18^m UT, еще до окончания затмения.

10. Покрытия звезд Луной

В течение суток Луна смещается по орбите примерно на 13°, двигаясь на фоне звезд с запада на восток. Движение Луны особенно заметно, когда ее путь пролегает вблизи ярких звезд. Иногда можно наблюдать впечатляющее явление: внезапно звезда исчезает у восточного края лунного диска и по прошествии некоторого времени вновь появляется с другой стороны. Невооруженному глазу доступно около тысячи звезд, которые могут быть покрыты Луной. Они располагаются в узкой полосе в 8° к северу и к югу от эклиптики.

Покрытие во многих отношениях сходно с солнечным затмением. Однако, поскольку лучи света от удаленной звезды можно рассматривать как параллельный пучок, лунная тень в этом случае имеет форму не конуса, а цилиндра с постоянным диаметром, равным диаметру самой Луны. Соответственно не существует и разделения на области тени и полутени. Лунная тень имеет размеры около четверти диаметра Земли и не может накрыть ее целиком, поэтому покрытие наблюдается в пределах определенной полосы на земной поверхности.

Луна не имеет атмосферы, поэтому падение блеска звезды, как и последующий подъем, происходит практически мгновенно. Моменты исчезновения и появления звезды нетрудно зафиксировать, что позволяет с большой точностью определить положение Луны. Поэтому покрытия звезд длительное время наблюдались с целью уточнения наших знаний о движении Луны и вращении Земли.

В многочисленных астрономических изданиях публикуются сведения, позволяющие заранее планировать наблюдения покрытий. Разумеется, сведения приводятся лишь для некоторых определенных пунктов, для перехода к другим пунктам наблюдения используются так называемые дифференциальные коэффициенты. Предвычисление покрытий не требует особо высокой точности, поэтому допустимы определенные упрощения. Например, можно не учитывать неправильности лимба, обусловленные рельефом лунной поверхности. Тем не менее предвычисление покрытий — трудоемкий процесс, поскольку из большого числа звезд необходимо отобрать те, которые в интересующий период времени могут быть покрыты Луной.

Программа `Occult` исследует возможность покрытия произвольно выбранных звезд в течение суток, шаг за шагом покрывая заданный период времени. Угловое расстояние между Луной и звездой минимально, когда их геоцентрическое прямое восхождение одинаково. Этот момент может быть определен простой итерацией. Если разница склонений Луны и звезды не слишком велика, значит, где-то на Земле может наблюдаться покрытие. Дальнейшее исследование движения лунной тени во время соединения позволит узнать, наблюдаемо ли покрытие в конкретном наблюдательном пункте.

Таблица 10.1. Сведения о некоторых звезд в Плеядах: номер по каталогу зодиакальных звезд, координаты для эпохи J2000 и собственные движения (за столетие) по каталогу PPM, визуальная звездная величина и название

ZC	α_{J2000}			μ_α	δ_{J2000}			μ_δ	m_{vis}	Название	
	h	m	s	s	°	′	″	″	m		
536	3	44	48,180	+0,06	+24	17	21,44	−5,1	5,4	Келено	16 Tau
537	3	44	52,532	+0,14	+24	6	48,01	−4,6	3,8	Электра	17 Tau
539	3	45	12,469	+0,07	+24	28	1,36	−5,8	4,4	Тайгета	19 Tau
541	3	45	49,566	+0,05	+24	22	3,63	−5,0	4,0	Майя	20 Tau
542	3	45	54,423	+0,04	+24	33	16,02	−4,6	5,9	Астеропа	21 Tau
545	3	46	19,537	+0,08	+23	56	53,42	−5,7	4,3	Меропа	23 Tau
552	3	47	29,073	+0,14	+24	6	18,38	−4,6	3,0	Альциона	η Tau
560	3	49	9,739	+0,13	+24	3	12,24	−4,7	3,8	Атлас	27 Tau
561	3	49	11,181	+0,09	+24	8	12,58	−4,3	5,2	Плейона	28 Tau

10.1. Видимые положения

Координаты звезд для вычислений покрытий можно найти в различных каталогах, например каталоге PPM (Position and Proper Motion Catalogue), каталоге SAO, публикуемомСмитсонианской астрофизической обсерваторией, и Каталоге зодиакальных звезд (каталог 3539 зодиакальных звезд для эпохи 1950.0), специально предназначенном для этих целей. Фрагмент каталога, содержащего наиболее важные сведения о звездах, принадлежащих к скоплению Плеяд, приведен в табл. 10.1. Наряду с прямым восхождением и склонением в нем содержатся значения собственных движений для сотни звезд. Понятно, что движение звезд в пространстве относительно Солнечной системы со временем приводит к изменению их координат на небесной сфере. Поэтому необходимо сперва скорректировать к дате наблюдений каталожные координаты (α_0, δ_0), приведенные для эпохи t_0, используя значения собственных движений (μ_α, μ_δ):

$$\begin{aligned} \alpha(t) &= \alpha_0 + \mu_\alpha(t - t_0), \\ \delta(t) &= \delta_0 + \mu_\delta(t - t_0). \end{aligned} \qquad (10.1)$$

Полученные координаты еще не могут быть использованы для вычисления покрытий. Видимые положения звезд связаны с положением точки весеннего равноденствия и небесного экватора в момент наблюдения и постоянно изменяются вследствие прецессии и нутации. Координаты звезд в каталогах приводятся для некоторой определенной эпохи, в настоящее время это преимущественно эпоха и равноденствие 2000 года, но некоторые каталоги по-прежнему содержат координаты для эпохи 1950, использовавшейся ранее.

Уравнения (2.14) и (2.15) позволяют вычислить усредненные долгопериодические изменения положения точки весеннего равноденствия. С их помощью единичный вектор, описывающий координаты звезды

$$e = \begin{pmatrix} \cos\delta \cos\alpha \\ \cos\delta \sin\alpha \\ \sin\delta \end{pmatrix}, \quad (10.2)$$

может быть преобразован от каталожной эпохи к эпохе наблюдения.

Нутация, вызванная изменениями в совокупном притяжении Солнца и Луны, накладывается на прецессию земной оси. Формулы для учета влияния нутации содержат долгопериодические члены $\Delta\varepsilon$ и $\Delta\psi$, достигающие абсолютных значений 9″ и 17″ соответственно (см. (6.14)). Истинные координаты звезд отличаются от их средних координат на эти величины. Необходимые поправки получают, используя формулы (6.15).

Аберрация, обусловленная конечностью скорости света, также вызывает изменения координат звезд. Явление аберрации состоит в том, что наблюдатель, движущийся вместе с Землей вокруг Солнца, видит звезду в направлении, несколько отличном от того, что видел бы наблюдатель, неподвижный относительно Солнца. В прямоугольных координатах видимое положение звезды e' может быть определено прибавлением к координатному вектору отношения скорости Земли к скорости света $v_{\oplus}/c$. После нормировки получаем:

$$e' = \frac{e + v_{\oplus}/c}{|e + v_{\oplus}/c|}. \quad (10.3)$$

Прямое восхождение и склонение звезды с учетом влияния аберрации находятся из этих скорректированных декартовых координат.

Взятые из каталога координаты могут быть полностью скорректированы на прецессию, нутацию и аберрацию с помощью следующей подпрограммы:

```
// Собственное движение
double RA  = RA_Cat  + mu_RA *(T_Epoch-T_CatEpoch),
double Dec = Dec_Cat + mu_Dec*(T_Epoch-T_CatEpoch);
// Переход от каталожной эпохи к истинному равноденствию даты
PN = NutMatrix(T)*PrecMatrix_Equ(T_CatEquinox, T);
// Precession, nutation
Vec3D e = PN * Vec3D(Polar(RA, Dec));
// Вектор скорости Земли (в единицах скорости света
v_Earth = Ecl2EquMatrix(T)*KepVelocity(Earth, T) / c_light,
// Аберрация
e = e + v_Earth;
e = e / Norm(e);
```

Для вычисления влияний прецессии и нутации использованы уже знакомые функции `PrecMatrix_Equ` и `NutMatrix`. Гелиоцентрическую скорость Земли вычисляет функция `KepVelocity`, использующая приближенную кеплерову орбиту и осуществляющая переход от эклиптических координат к экваториальным.

Чтобы использовать звездный каталог для предвычислений покрытий, применяется отдельный класс Star:

```
// Класс Star
//
class Star {
  public:

    static void SetCatEquinox( double T );
    static void SetCatEpoch( double T ).
    static void SetEpoch( double T ):

    Vec3D  Apparent().  // Вычисление видимого положения звезды

    inline char*  Name() { return name: }.
    inline double Mag() { return mag: }:

    friend istream& operator >> (istream& is. Star& aStar).

  private·

    double RA_Cat:               // Каталожное прямое восхождение в рад
    double Dec_Cat:              // Каталожное склонение в рад
    double mu_RA:                // Собственное движение по прямому восхождению
                                 // за столетие в рад
    double mu_Dec:               // Собственное движение по склонению за столетие в рад
    double mag:                  // Визуальная звездная величина
    char   name[12]:             // Название

    static double T_CatEquinox:  // Равноденствие каталога
    static double T_CatEpoch.    // Эпоха каталога
    static double T_Epoch:       // Текущая эпоха

    static Mat3D  PN.            // Поправка за прецессию и нутацию
    static Vec3D  v_Earth.       // Вектор скорости Земли

}.
```

Этот класс создает оператор ввода, при помощи которого координаты, собственное движение, звездная величина и название звезды считываются из текстового файла и преобразуются в форму, пригодную для вычислений. Неизменные величины, общие для всех объектов каталога, такие как эпоха и равноденствие, вводятся следующим образом:

```
void Star:·SetCatEquinox (double T)
{
  T_CatEquinox = T:
}


void Star·:SetCatEpoch (double T)
{
  T_CatEpoch = T:
}
```

Текущая эпоха определяется следующим образом:

```
void Star::SetEpoch (double T)
{
  T_Epoch = T;  // Сохранение эпохи
PN = NutMatrix(T)*PrecMatrix_Equ(T_CatEquinox, T);
v_Earth = Ecl2EquMatrix(T)*KepVelocity(Earth, T) / c_light;
}
```

Этими средствами устанавливаются значения прецессии и нутации, а также гелиоцентрической скорости Земли, одинаковые для всех звезд. Полученные величины используются для вычислений видимых положений звезд методом Apparent:

```
Vec3D Star.:Apparent()
{
double RA  = RA_Cat  + mu_RA *(T_Epoch-T_CatEpoch);
  double Dec = Dec_Cat + mu_Dec*(T_Epoch-T_CatEpoch);
Vec3D e = PN * Vec3D(Polar(RA, Dec));
e = e / Norm(e);

  return e;
};
```

Массив объектов из класса Star вводится в программу Occult функцией ReadCatalogue:

```
//------------------------------------------------------------------------------
// ReadCatalogue: считывание каталожных координат звезд из файла ввода
//
// Ввод:
//   Filename     имя каталожного файла, подлежащего считыванию
//
// Вывод:
//   Stars        массив звезд, размещаемых в каталоге
//   StarsRead    Number of stars read from catalogue file - Число звезд,
//                считанных из каталожного файла
//
// Замечание: память, выделенная для массива Stars, очищается
// вызывающей процедурой
//------------------------------------------------------------------------------
void ReadCatalogue (char* Filename, Star*& Stars, int& StarsRead)
{
  //
  // Variables
  //
  ifstream inp;
  double   Y_Epoch, Y_Eqx;
  int      i,

  inp.open(Filename);

  // Считывание и размещение эпохи и равноденствия
  inp >> Y_Epoch >> Y_Eqx;

  Star::SetCatEpoch((Y_Epoch-2000.0)/100.0);
  Star::SetCatEquinox((Y_Eqx-2000.0)/100.0);
```

```
  // Считывание координат звезд
  StarsRead = 0.
  inp >> StarsRead: inp.ignore(81.'\n'):
  Stars = new Star[StarsRead].  // Выделение памяти для массива из каталога

  if (Stars != NULL) {
    for (i=0: i<StarsRead. i++)  inp >> Stars[i].
  }
  else
    cerr << "Insufficient memory to read star catalogue." << endl:

  inp.close():

  cout << endl << " " << StarsRead
       << " stars read from catalogue" << endl << endl.
}
```

10.2. Геоцентрическое соединение

Для предсказания покрытий необходимо из большого числа звезд близ эклиптики отобрать те, которые в заданный период времени могут быть покрыты Луной. Для этого прежде всего следует определить момент времени, когда геоцентрическое прямое восхождение α_M Луны приближается к значению прямого восхождения звезды $\alpha_\star$. Сравнение склонений в момент соединения позволит определить, будет ли где-либо на Земле наблюдаться покрытие. Время соединения является исходной точкой для дальнейших более точных вычислений возможных покрытий, которые будут обсуждаться в следующих разделах.

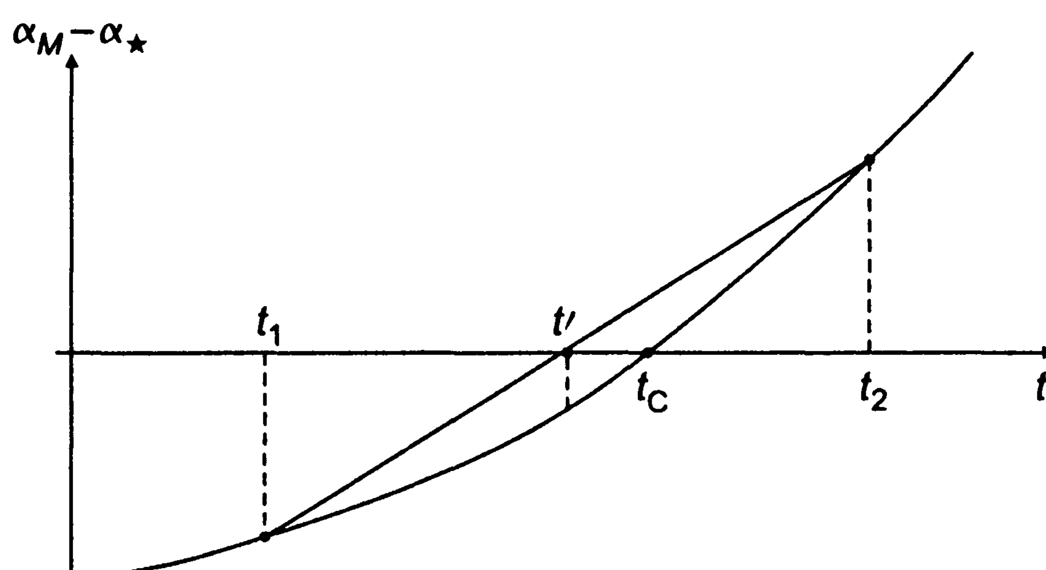


Рис. 10.1. Вычисление момента соединения методом итерации

В интервале времени от t_1 до t_2 прямое восхождение Луны изменяется от $\alpha_M(t_1)$ до $\alpha_M(t_2)$. Если прямое восхождение звезды $\alpha_\star$попадает в этот интервал, время соединения можно определить с помощью линейной интерполяции (рис. 10.1). Поскольку прямое восхождение Луны возрастает достаточно равномерно, допустимо следующее приближение:

$$t_c \approx t' = t_2 - \Delta\alpha_2 \cdot \frac{t_2 - t_1}{\Delta\alpha_2 - \Delta\alpha_1},$$

где $\Delta\alpha_i = \alpha_M(t_i)\ \alpha_\star$. В зависимости от того, будет ли полученное прямое восхождение Луны больше или меньше прямого восхождения звезды, заменим t_1 или t_2 на полученное значение t'. Тогда неравенство $\alpha_M(t_1) \leq \alpha_\star \leq \alpha_M(t_2)$ остается верным и можно повторить процедуру для достижения большей точности. Благодаря равномерности движения Луны, эта процедура, известная под названием regula falsi, быстро приводит к желаемому результату. Небольшое изменение позволяет ускорить сходимость процедуры. В этом случае после вычисления t' и $\Delta\alpha' = \alpha_M(t') - \alpha_\star$, начальные параметры для последующей итерации выбираются следующим образом:

- если $\Delta\alpha'\Delta\alpha_2 \leq 0$, то заменить $(t_1, \Delta\alpha_1)$ на $(t_2, \Delta\alpha_2)$;
- если $\Delta\alpha'\Delta\alpha_2 > 0$, то заменить $(t_1, \Delta\alpha_1)$ на $(t_1, \Delta\alpha_1\Delta\alpha_2/(\Delta\alpha_2 + \Delta\alpha'))$.

Также в обоих случаях $(t_2, \Delta\alpha_2)$ заменяется на $(t', \Delta\alpha')$. В этом варианте regula falsi, известном как метод Pegasus, верхняя граница исследуемого интервала не фиксирована, что значительно сокращает объем вычислений.

```
//------------------------------------------------------------------------------
//
// Pegasus: Нахождение решения методом Pegasus
//   PegasusFunct  Указатель исследуемой функции
//   LowerBound    Нижняя граница исследуемого интервала
//   UpperBound    Верхняя граница исследуемого интервала
//   Accuracy      Желаемая точность решения
//   Root          Решение найдено (действительно, только если процедура успешно выполнена)
//   Success       Процедура успешно выполнена
// Примечание.
//   Подразумевается, что решение должно находиться в интервале
//   [LowerBound, UpperBound], границы которого могут иметь разные знаки
//
//------------------------------------------------------------------------------
void Pegasus ( PegasusFunct f,
               double LowerBound, double UpperBound, double  Accuracy,
               double& Root, bool& Success  );
{
  const int MaxIterat = 30;

  //
  // Переменные
  //
  double x1 = LowerBound; double f1 = f(x1);
  double x2 = UpperBound, double f2 = f(x2),
  double x3 = 0.0;        double f3 = 0.0,

  int Iterat = 0;


  // Initialization
  Success = false;
  Root    = x1,


  // Итерация
  if ( f1 * f2 < 0.0 )
    do
```

```
    {
      // Приближенное решение методом интерполяции
      x3 = x2 - f2/( (f2-f1)/(x2-x1) ); f3 = f(x3);

      // Замена (x1,f2) и (x2,f2) на новые значения, так чтобы решение по-прежнему
      // находилось в интервале [x1,x2]
      if ( f3 * f2 <= 0.0 ) {
        // Root in [x2,x3]
        x1 = x2; f1 = f2; // Замена (x1,f1) на (x2,f2)
        x2 = x3; f2 = f3; // Замена (x2,f2) на (x3,f3)
      }
      else {
        // Корень в интервале [x1,x3]
        f1 = f1 * f2/(f2+f3); // Замена (x1,f1) на (x1,f1')
        x2 = x3; f2 = f3;     // Замена (x2,f2) на (x3,f3)
      }

      if (fabs(f1) < fabs(f2))
        Root = x1;
      else
        Root = x2;

      Success = (fabs(x2-x1) <= Accuracy);
      Iterat++;
    }
    while ( !Success && (Iterat<MaxIterat) );
}
```

Приведенная процедура Pegasus определяет точки, в которых функция f(x)

```
// Прототип функции Pegasus: double f(double x);
typedef double (*PegasusFunct) (double x);
```

обращается в нуль.

В процессе определения момента соединения и в других стадиях вычислений приходится перебирать целый ряд значений положения Луны. Для этого лучше использовать полином Чебышева, как было описано выше. По нескольким точно вычисленным положениям Луны можно построить простое выражение для лунной орбиты, которое с помощью нескольких операций может быть распространено на любой момент времени. Наивысший порядок полинома (например, десять) выбирается так, чтобы можно было с достаточной точностью получить координаты Луны в течение одних суток. Как было сказано в главе 8, для этого достаточно объявить объект класса Cheb3D в функции MoonEqu. Соответственно вектор экваториальных координат Луны может быть определен методом Value(T). Прямое восхождение получается из него с помощью векторного оператора [phi]. Полученные координаты относятся к истинному равноденствию даты, то есть к исправленному на прецессию и нутацию положению точки весеннего равноденствия и небесного экватора.

```
Const double Interval = 1.0/36525.0;    // 1d [cy]
Cheb3D ChebMoonEqu(MoonEqu, 10, Interval);
double RA_Star,

//------------------------------------------------------------------------
//
// RA_Diff: Функция итерации для определения момента соединения
```

```
//
//   T - число столетий, прошедших с эпохи J2000
//
// Возвращаемое значение:  Разница прямых восхождений Луны и звезды в рад
//
//------------------------------------------------------------------------
double RA_Diff (double T)
{
  Vec3D r_Moon = ChebMoonEqu.Value(T);    // Lunar position at time T
  return Modulo ( r_Moon[phi] - RA_Star + pi, pi2 ) - pi;
}
```

Момент соединения может быть определен для конкретной звезды лишь при условии, что в рассматриваемый период времени $\Delta\alpha(T)$ не изменяет знак.

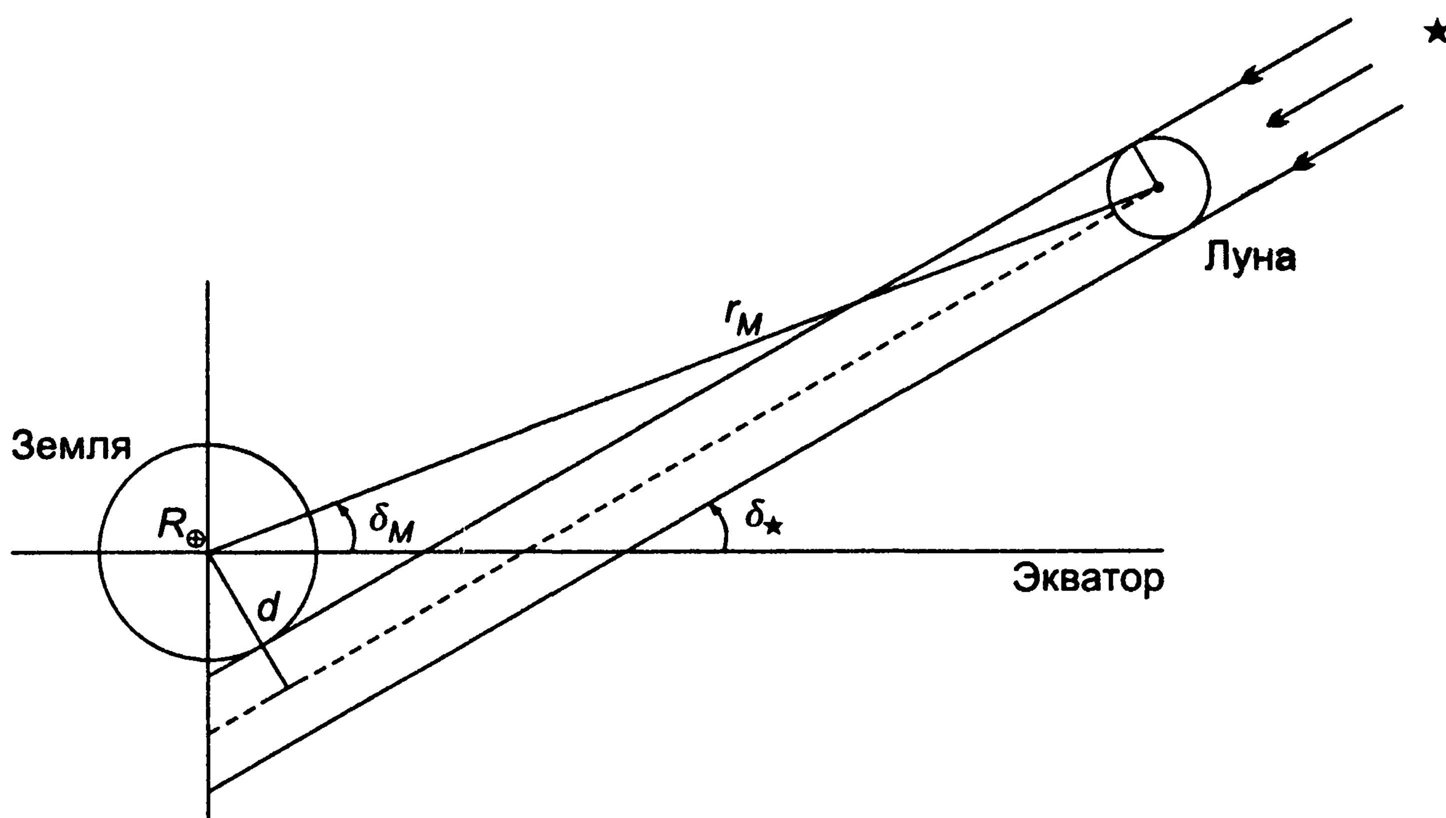


Рис. 10.2. Положение Луны в момент соединения по прямому восхождению

Теперь возможность покрытия можно определить по координатам Луны в момент соединения. Рисунок 10.2 поясняет то обстоятельство, что лунная тень касается Земли лишь тогда, когда склонения Луны и звезды не сильно различаются. Расстояние $d < R_\oplus + R_M \approx 1{,}3R_\oplus$.

Поэтому в момент соединения по прямому восхождению должно выполняться условие:

$$d(t_c) = r_M \cdot |\sin(\delta_M - \delta_\star)| = |\boldsymbol{r}_M - (\boldsymbol{e} \cdot \boldsymbol{r}_M)\boldsymbol{e}| < R_\oplus + R_M \approx 1{,}3R_\oplus.$$

С учетом наклонения лунной орбиты к небесному экватору значение $d(tc)$ не должно превышать минимальное расстояние между лунной тенью и центром Земли более чем на 0,2 земных радиуса. Поэтому вероятные покрытия могут быть отобраны по следующему критерию:

$$|\boldsymbol{r}_M - (\boldsymbol{e} \cdot \boldsymbol{r}_M)\boldsymbol{e}| \begin{Bmatrix} < \\ > \end{Bmatrix} 1{,}5R_\oplus \Rightarrow \begin{Bmatrix} \text{покрытие возможно} \\ \text{покрытие невозможно} \end{Bmatrix}. \qquad (10.4)$$

Итак, первая часть нашей программы для вычисления покрытий, `Conjunct`, может быть теперь составлена. Она вначале отбирает звезды, прямое восхождение ко-

торых Луна проходит в промежутке между двумя заданными моментами времени T_1 и T_2. Далее определяется момент соединения и исследуется возможность касания лунной тенью поверхности Земли. Если условия возможности покрытия выполняются, найденный момент соединения возвращается вызывающей программе.

```
//------------------------------------------------------------------------
// Conjunct: Исследование возможности покрытия
//
// Ввод·
//   T1       Начало исследуемого промежутка
//   T2       Конец исследуемого промежутка
//   RA1      Прямое восхождение Луны в момент T1 в рад
//   RA2      Прямое восхождение Луны в момент T2 в рад
//   e        Направление на звезду (экваториальные координаты)
//   Conj     Указатель, обозначающий, что покрытие по крайней мере вероятно
//   T_Conj   Момент соединения
//------------------------------------------------------------------------
void Conjunct ( double T1, double T2, double RA1, double RA2, Vec3D& e,
                bool& Conj, double& T_Conj )
{
const double eps = Rad*1.0e-4;        // Заданная точность по прямому восхождению
Vec3D r_Moon,
  T_Conj = 0.0,                       // Значение по умолчанию
  bool Success = false,
if ( RA2<RA1 ) RA2+=pi2;              // Отображение в [RA1,RA1+2*pi[

  RA_Star = e[phi];
  if ( RA_Star<RA1 ) RA_Star+=pi2,    // Отображение в [RA1,RA1+2*pi[

  Conj = ( (RA1<=RA_Star) && (RA_Star<=RA2) );

  if ( Conj ) {
    Pegasus(RA_Diff, T1, T2, eps, T_Conj, Success),
    r_Moon = ChebMoonEqu.Value(T_Conj),
    Conj = ( Norm( r_Moon-e*Dot(r_Moon,e) ) < 1.5*R_Earth ),
  }
}
```

10.3. Главная плоскость

Сечение цилиндрической лунной тени плоскостью, перпендикулярной к оси последней, представляет собой круг, диаметр которого равен диаметру самой Луны. Такая плоскость, проходящая через центр Земли, имеет название *главной плоскости*. Относительные положения лунной тени и наблюдателя могут быть заменены на их координаты в проекции на главную плоскость.

Поместим центр Земли в начале прямоугольной системы координат с осью z, направленной на звезду с координатами $(\alpha_\star, \delta_\star)$, и осью x, лежащей в плоскости земного экватора (рис. 10.3). В таком случае оси x и y образуют главную плоскость. Соответствующий единичный вектор

$$e_x = \frac{e_3 \times e}{|e_3 \times e|}, \qquad e_y = e \times e_x, \tag{10.5}$$

где $\boldsymbol{e} = \boldsymbol{e}_z$ — единичный вектор, направленный на звезду, и $\boldsymbol{e}_3 = (0, 0, 1)^T$ — единичный вектор, направленный на Северный полюс мира. Проекция вектора $\boldsymbol{r}_M$ экваториальных координат Луны на главную плоскость имеет координаты:

$$\begin{aligned} x_M &= \boldsymbol{e}_x \cdot \boldsymbol{r}_M, \\ y_M &= \boldsymbol{e}_y \cdot \boldsymbol{r}_M, \end{aligned} \qquad (10.6)$$

являющиеся координатами центра лунной тени в главной плоскости.

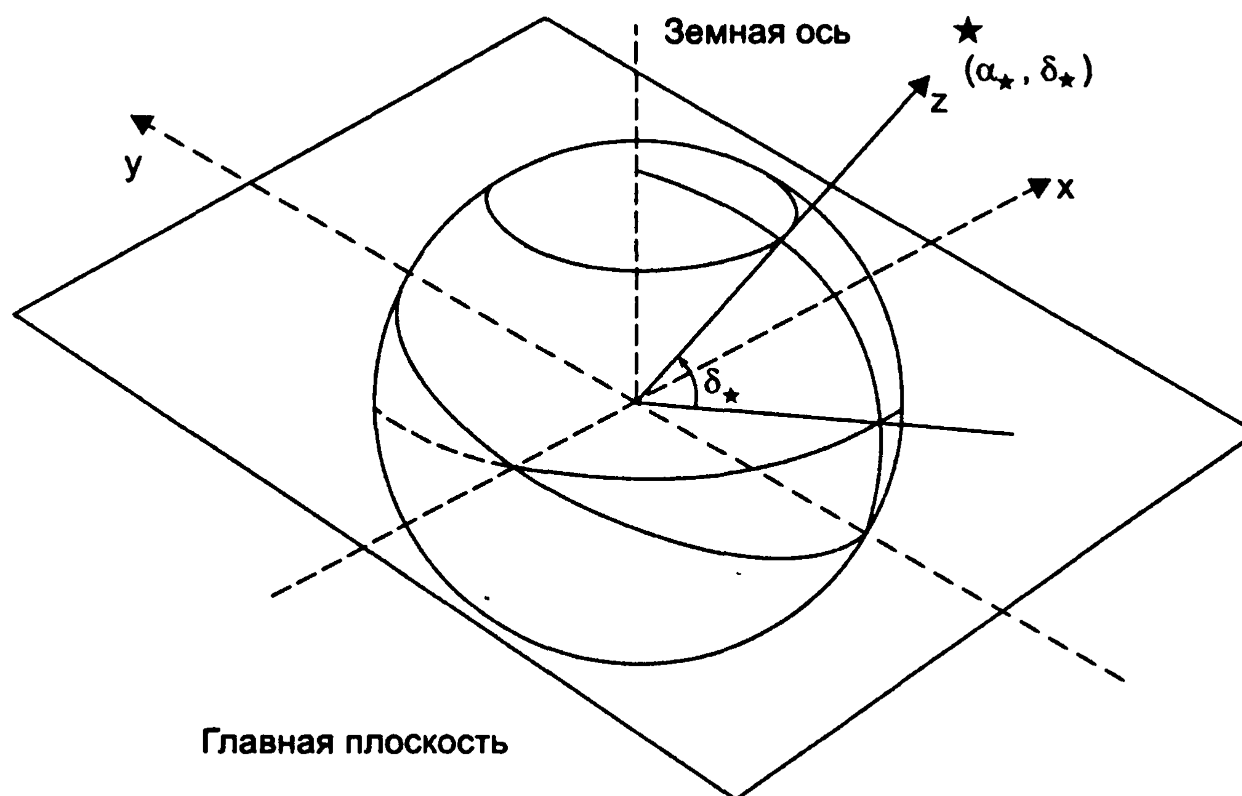


Рис. 10.3. Главная плоскость

Таким же образом экваториальные координаты наблюдателя $\boldsymbol{r}_O$ дают точку (x_O, y_O) на главной плоскости, представляющую проекцию луча, направленного от звезды «сквозь» наблюдателя. «Склонение» точки наблюдения равняется углу между позиционным вектором — вектором, направленным к ней из центра Земли, и плоскостью земного (и небесного соответственно) экватора. Этот угол известен под названием *геоцентрической* широты φ', которое не следует путать с *географической* широтой φ. Последняя величина, гораздо чаще используемая, представляет собой угол между земной осью и горизонтом в пункте наблюдения и равняется высоте Полярной звезды над горизонтом[1]. Истинная форма вращающейся Земли отличается от шара и представляет собой эллипсоид вращения, поэтому возникает различие между величинами φ и φ' (см. рис. 9.5). Расстояние от центра Земли до полюсов примерно на 20 км меньше ее экваториального радиуса $r_\oplus = 6378{,}14$ км. Отношение разности между экваториальным и полярным радиусом к величине экваториального радиуса дает величину сжатия Земли $f \approx 1/298$ [2]. Обычно геоцентрическая широта не известна, и ее следует вычислять, исходя из географической широты места, которая может быть найдена в атласах или соответствующих изданиях.

---

[1] Полярная звезда находится не точно в Полюсе мира, а примерно в градусе от него. — *Примеч. перев.*

[2] Символ f используется далее для обозначения одной из двух координат в главной плоскости, однако это не должно смущать читателя.

Функция Site использует формулу (9.9) для вычисления декартовых координат наблюдателя относительно земного экватора и гринвичского меридиана:

$$r_O^G = \begin{pmatrix} r\cos\lambda\cos\varphi' \\ r\sin\lambda\cos\varphi' \\ r\sin\varphi' \end{pmatrix} = \frac{R_\oplus}{\sqrt{1-e^2\sin^2\varphi}} \begin{pmatrix} \cos\lambda\cos\varphi \\ \sin\lambda\cos\varphi \\ (1-e^2)\sin\varphi \end{pmatrix}. \tag{10.7}$$

Позиционный вектор относительно экватора и точки равноденствия определяется по формуле:

$$r_O = R_Z(-\Theta_0(t))r_O^G, \tag{10.8}$$

где $\Theta_0(t)$ — звездное время по Гринвичу. Звездное время может быть вычислено с помощью функции GMST, описанной в разделе 3.3.

```
//---------------------------------------------------------------------------
// Site· Вычисление геоцентрических координат точки на земной поверхности
//
//   lambda    Географическая долгота (направление на восток положительно) в рад
//   phi       Географическая долгота в рад
//
// Возвращаемые значения·   Геоцентрические координаты в км
//---------------------------------------------------------------------------
Vec3D Site (double lambda. double phi)
{
  // Константы
  //
  const double f     = 1 0/298 257:   // Сжатие Земли
  const double e_sqr = f*(2.0-f):     // Квадрат эксцентриситета
  const double cos_phi = cos(phi):    // Косинус и синус географической широты
  const double sin_phi = sin(phi):

  // Переменные
  //
  double  N = R_Earth / sqrt (1.0-e_sqr*(sin_phi*sin_phi)).

  // Позиционный вектор в прямоугольных координатах (км)
  //
  return Vec3D ( N*cos_phi*cos(lambda).
                 N*cos_phi*sin(lambda).
                 (1.0-e_sqr)*N*sin_phi  ).
}
```

Здесь необходимо вернуться к рассмотрению различных систем счета времени. При вычислении положения Луны предполагалось, что все моменты приводятся по равномерному[1] эфемеридному времени (Ephemeris Time ET = Dynamic Time TDB/TDT). Эта система счета времени обычно используется при вычислениях орбит небесных тел, что и обусловливает ее название. Однако при вычислении звездного времени (при помощи функции LMST — см. раздел 3.3) требуется знание Всемирного времени (Universal Time, UT). Разница между ET и UT в настоящее время составляет около минуты, и ее значение можно взять из табл. 3.1.

[1] В отличие от солнечного или звездного времени, измеряемого по вращению Земли, обладающему неравномерностью. — *Примеч. перев.*

Функция ETminUT (см. раздел 9.3) вычисляет ее величину для периода с 1825 по 2005 годы. Ее значение на последующие годы приводится в различных астрономических изданиях. Поскольку разница между Всемирными эфемеридным временем изменяется очень медленно, достаточно вычислить ее один раз в начале нашей программы.

10.4. Покрытия и открытия

Обстоятельства покрытия определяются значением разности координат Луны и наблюдателя в главной плоскости:

$$f = x_M - x_O,$$
$$g = y_M - y_O.$$

Если расстояние от наблюдателя до центра лунной тени $\sqrt{f^2 + g^2}$ меньше радиуса Луны R_M, наблюдатель увидит покрытие. Моменты покрытия и открытия определяются равенством:

$$f(t)^2 + g(t)^2 = R_M^2 = k^2 \cdot R_\oplus^2, \qquad (10.9)$$

где $k = R_M/R_\oplus = 0{,}2725$ — отношение радиусов Луны и Земли.

В своем движении по орбите Луна покрывает расстояние, равное диаметру Земли, примерно за четыре часа. В течение этого времени лунная тень движется по земной поверхности. Вычисление контактов следует начинать с момента времени, предшествующего соединению примерно на два с половиной часа. Для определения моментов, удовлетворяющих условию (10.9), величина

$$s(t) = f^2(t) + g^2(t) - k^2 R_\oplus^2$$

вычисляется с шагом по времени в 15^m. Вначале s положительно, поскольку место наблюдения находится вне лунной тени. Когда s меняет знак, звезда скрывается за лунным лимбом. По трем последовательным значениям s_-, s_0 и s_+ можно построить параболу, отражающую ход изменения $s(t)$ в течение получаса. Если в этот период s единожды или дважды обращается в нуль, эти точки находят, решая квадратное уравнение, если же нет — берут следующие три значения s и повторяют вычисления. Эта процедура в точности та же, что использовалась для вычисления моментов восхода и захода: функция Quad, представленная в главе 3, по трем заданным точкам осуществляет квадратичную интерполяцию и находит точки, в которых функция обращается в нуль. Определение моментов покрытий и открытий осуществляет программа Examine, полный текст которой приведен в конце настоящей главы.

Предвычисление покрытий позволяет получить помимо времени контактов и другие интересующие наблюдателя сведения: позиционный угол точек покрытия и открытия, а также дифференциальные коэффициенты. Способ определения позиционного угла показан на рис. 10.4. Это угол между направлением на север и линией, проведенной из центра лунного диска к точке на лимбе, где происходит покрытие или открытие. Позиционный угол всегда отсчитывается против часо-

вой стрелки от 0° до 360°. Величина позиционного угла[1] θ зависит только от координат f и g в главной плоскости в момент покрытия или открытия. Он может быть получен из следующих соотношений:

$$\cos\vartheta = -g/\sqrt{f^2+g^2}, \quad \sin\vartheta = f/\sqrt{f^2+g^2}. \tag{10.10}$$

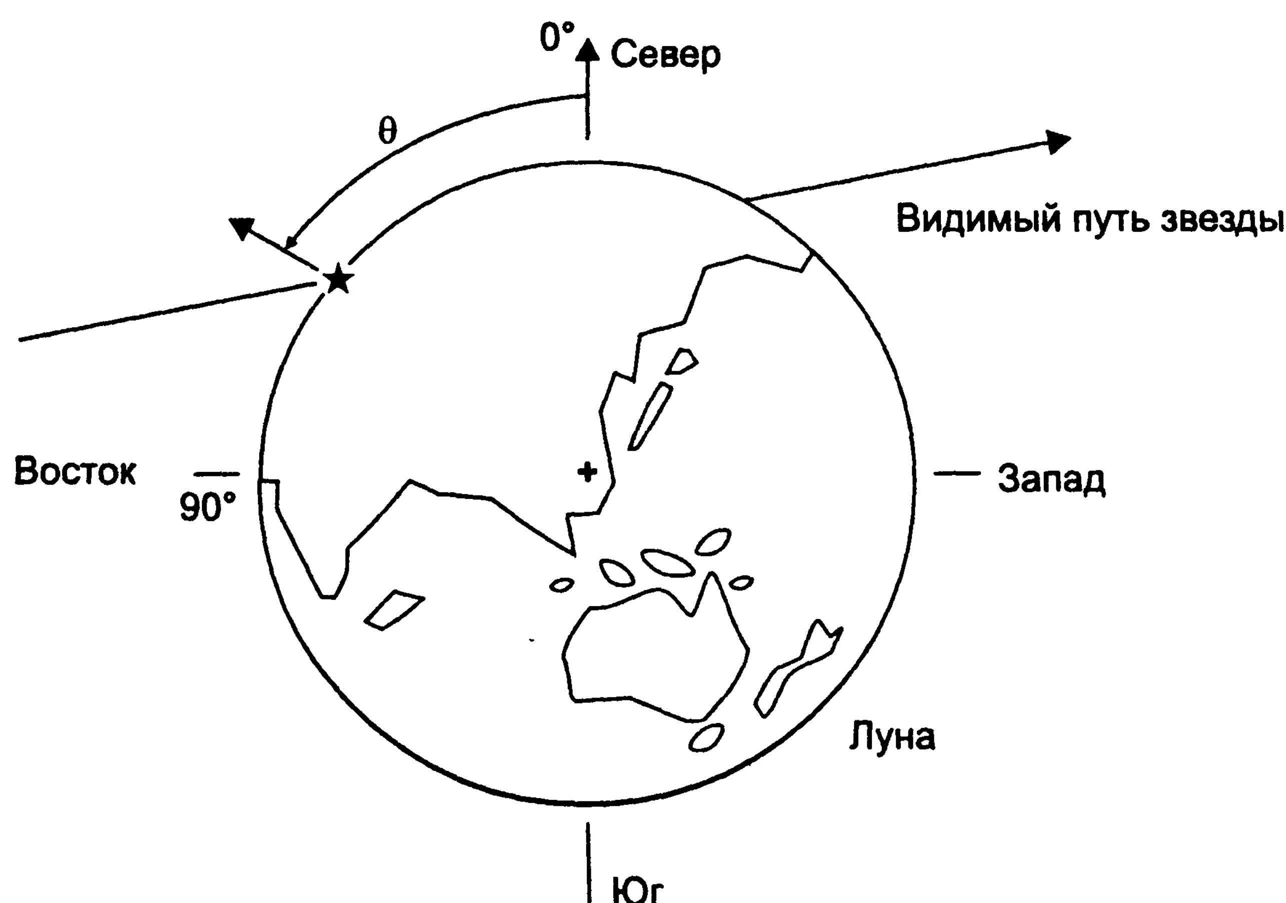


Рис. 10.4. Позиционный угол звезды в момент контакта с лунным лимбом

Дифференциальные коэффициенты — величины, позволяющие пересчитать моменты контактов, вычисленные для одного избранного пункта, для другого близкого пункта наблюдений. Коэффициент a показывает, насколько время контакта зависит от географической долготы λ, а b соответственно — от географической широты φ:

$$a = \frac{dt}{d\lambda}, \qquad b = \frac{dt}{d\varphi}.$$

Типичные значения обоих коэффициентов составляют около $1^m/°$. Для касательных покрытий, когда моменты исчезновения и появления звезды следуют один за другим, эти коэффициенты резко возрастают[2]. Рассмотрев выражение (10.9) с позиций связи между временем контакта t и положением наблюдателя $\boldsymbol{r}$, можно заметить, что производная от t может быть выражена через Декартовы координаты наблюдателя:

$$\frac{dt}{d\boldsymbol{r}} = \frac{f\boldsymbol{e}_x + d\boldsymbol{e}_y}{f\dot{f} + g\dot{g}}. \tag{10.11}$$

[1] В отечественной литературе обычно употребляется обозначение P для позиционного угла. — *Примеч. перев.*

[2] Настолько, что интерполяция для соседних пунктов становится невозможной. — *Примеч. перев.*

Здесь $\dot{f}$ и $\dot{g}$ — производные по времени координат f и g, которые для простоты можно представить в следующем виде:

$$\dot{f}(t) \approx \frac{f(t+0{,}25^{\mathrm{h}}) - f(t)}{0{,}25^{\mathrm{h}}}, \qquad \dot{g}(t) \approx \frac{g(t+0{,}25^{\mathrm{h}}) - g(t)}{0{,}25^{\mathrm{h}}}.$$

Дифференцируя позиционный вектор по географической долготе и широте

$$\frac{d\boldsymbol{r}}{d\lambda} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \boldsymbol{r}, \qquad \frac{d\boldsymbol{r}}{d\varphi} = \boldsymbol{r} \times \frac{d\boldsymbol{r}}{d\varphi} \Big/ \left|\frac{d\boldsymbol{r}}{d\lambda}\right|, \tag{10.12}$$

можно выразить искомые коэффициенты следующим образом:

$$a = \frac{dt}{d\boldsymbol{r}} \frac{d\boldsymbol{r}}{d\lambda}, \qquad b = \frac{dt}{d\boldsymbol{r}} \frac{d\boldsymbol{r}}{d\varphi}. \tag{10.13}$$

При вычислении этих коэффициентов можно пренебречь разницей между географической (φ) и геоцентрической (φ') широтой.

10.5. Программа Occult

Программа Occult вычисляет покрытия звезд Луной, наблюдаемые в заданном пункте в течение интересующего промежутка времени. Вместе с моментами покрытий и открытий определяются также соответствующие им позиционные углы и дифференциальные коэффициенты, позволяющие осуществить переход к соседнему пункту наблюдения.

Координаты интересующих звезд должны быть внесены в файл данных Occult.dat. Первая строка этого файла содержит эпоху и равноденствие подходящего каталога (обычно 1950.0 или 2000), а также число исследуемых звезд. После учета собственных движений звезд и приведения их координат к истинному равноденствию для каждой звезды получается строка, содержащая:

- прямое восхождение (в $^{\mathrm{h\ m\ s}}$),
- собственное движение по прямому восхождению за столетие (в $^{\mathrm{s}}$),
- склонение (в ° ′ ″),
- собственное движение по склонению за столетие (в ″),
- название звезды.

Эти величины приводятся в формате

```
_hh_mm_ss.sss__ss.ss___+dd mm ss.ss__sss.s__mm.m___ccccccccccc
```

Название звезды в этой строке ограничено одиннадцатью знаками, однако изменение в классе Star позволяет в любое время его расширить. Ниже приводится пример структуры файла данных для различных звезд Плеяд (см. табл. 10.1).

```
2000.0 2000.0 9
3  44 48.180   0.06   +24 17 21.44   -5.1   5.4   Celaeno
3  44 52.532   0.14   +24  6 48.01   -4.6   3.8   Electa
3  45 12.469   0.07   +24 28  1.36   -5.8   4.4   Taygeta
3  45 49.566   0.05   +24 22  3.63   -5.0   4.0   Maia
3  45 54.423   0.04   +24 33 16.02   -4.6   5.9   Asterope
3  46 19.537   0.08   +23 56 53.42   -5.7   4.3   Merope
3  47 29.073   0.14   +24  6 18.38   -4.6   3.0   Alcyone
3  49  9.739   0.13   +24  3 12.24   -4.7   3.8   Atlas
3  49 11.181   0.09   +24  8 12.58   -4.3   5.2   Pleione
```

В начале работы программа запрашивает географические координаты пункта наблюдений и интересующий период времени. Для этого периода с шагом в одни сутки определяются полиномы Чебышева. После вычисления видимых положений звезд, содержащихся в файле данных, функция Examine исследует возможность их покрытий. Упоминавшаяся процедура Conjunct определяет время соединения Луны и звезды, а также расстояние от центра Земли до оси лунной тени. Если покрытие возможно, Examine вычисляет моменты покрытия и открытия для заданного пункта наблюдений, после чего программа Contact рассчитывает соответствующие позиционный угол и дифференциальные коэффициенты для каждого контакта.

Программа выводит сведения лишь о тех покрытиях, которые происходят при благоприятных условиях видимости: Contacts анализирует высоту звезды и Солнца над горизонтом, фазу Луны, освещенность лунного лимба и блеск покрываемой звезды. Для этого используются следующие критерии:

- В момент покрытия или открытия звезда должна находиться на высоте не менее 5° над горизонтом. Для звезд ярче $1^m_{\cdot}9$ это ограничение снижается до 2°.
- Звезда должна быть не слабее $7^m_{\cdot}5$.
- В момент покрытия или открытия Солнце должно находиться по крайней мере на 6° градусов ниже горизонта (гражданские сумерки). Для звезд ярче $5^m_{\cdot}5$ это ограничение снижается до 3°, для звезд блеском от $2^m_{\cdot}0$ до $4^m_{\cdot}0$ приводятся все покрытия, происходящие от захода до восхода Солнца, а для более ярких — даже в дневное время.
- При угле фазы меньше 12°, то есть в пределах 24 часов до или после полнолуния, предельная величина звезды составляет $3^m_{\cdot}5$, при угле до 24° — $5^m_{\cdot}5$ и при угле до 36° — $6^m_{\cdot}5$ соответственно.
- Рассматриваются покрытия ярким краем лунного диска звезд не слабее $4^m_{\cdot}5$ и открытия — звезд не слабее $3^m_{\cdot}5$. При выходе из-за темного края Луны предельная величина звезд установлена в $6^m_{\cdot}5$.

Необходимо отметить, что в выходных данных покрытия приводятся не обязательно в хронологическом порядке, но в зависимости от расположения объектов в файле данных.

```
//------------------------------------------------------------------------
// Модуль:     Программа Occult (Occult.cpp)
//
// Задачи программы:    Предвычисление покрытий звезд Луной
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
```

```
//------------------------------------------------------------------------
#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Cheb.h"
#include "APC_Const.h"
#include "APC_IO.h"
#include "APC_Math.h"
#include "APC_Moon.h"
#include "APC_Phys.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time.h"
#include "APC_VecMat3D.h"

using namespace std;

// Константы
//
const double Interval  = 1.0/36525.0;    // 1d [cy]

//
// Основные объекты и переменные
//
Cheb3D ChebMoonEqu(MoonEqu, 10, Interval);
double RA_Star,

//
// Класс Star
//
class Star {
  ...
}
//
// Класс Event
//

enum enEvent {dummy=-1, in=0, out=1};  // определение покрытий и открытий

class Event {

  public:

    Event();
    Event( const Star& aStar, double MjdUT, enEvent InOut, bool visible,
           double PosAng, double a, double b );

    inline bool IsVisible() { return m_visible; };

    friend ostream& operator << (ostream& os, Event& anEvent);

  private:

    bool      m_visible;
    Star      m_Star;
    double    m_MjdUT;
```

```
    enEvent   m_InOut;
    double    m_PosAng;
    double    m_a,m_b;
};

//
// Constructors
//
Event::Event()
: m_visible(false),
  m_InOut(dummy)
{
}

Event::Event( const Star& aStar, double MjdUT, enEvent InOut, bool visible,
              double PosAng, double a, double b )
 : m_visible(visible),
   m_Star(aStar),
   m_MjdUT(MjdUT),
   m_InOut(InOut),
   m_PosAng(PosAng),
   m_a(a),
   m_b(b)
{
}

//
// Оператор вывода
//
ostream& operator << (ostream& os, Event& anEvent)
{
  if ( anEvent.m_InOut != dummy ) {

    // Вывод сообщений о покрытиях, удовлетворяющих критериям наблюдений
    os << " " << DateTime(anEvent.m_MjdUT)
       << "  " << anEvent.m_Star.Name()
       << fixed << setprecision(1)
       << setw(5) << anEvent.m_Star.Mag()
       << "    " << ( (anEvent.m_InOut==in)? "D" : "R" )
       << "    " << Time(24.0*Modulo(anEvent.m_MjdUT, 1.0),HHMMSS)
       << fixed << setprecision(1)
       << setw(7) << int(Deg*anEvent.m_PosAng + 0.5)
       << setw(8) << anEvent.m_a/Deg
       << setw(7) << anEvent.m_b/Deg
       << endl;
  }

  return os;
}

...

//------------------------------------------------------------------------
//
//   GetInput  Запросы пользователю на ввод данных
//   MjdStart  Начало периода, исследуемого на возможность покрытий
//   MjdEnd    Конец периода, исследуемого на возможность покрытий
//   ET_UT     Разность между эфемеридным и Всемирным временем, с
//   R_Obs     Геоцентрические координаты наблюдателя, км
//
// Note: MjdStart and MjdEnd as Modified Julian Dates
```

```
//
//-------------------------------------------------------------------------------
void GetInput ( double& MjdStart, double& MjdEnd,  double& ET_UT, Vec3D& R_Obs )
{
int     year,month,day;
  bool    valid;
  double  Lambda, Phi;

  // Запрос об исследуемом интервале
  cout << " Period of time for prediction of occultations" << endl;

  cout << "  first date (yyyy mm dd)                    ... ";
  cin >> year >> month >> day; cin.ignore(81,'\n');
  MjdStart = Mjd(year,month,day);

  cout << "  last  date (yyyy mm dd)                    ... ";
  cin >> year >> month >> day; cin.ignore(81,'\n');
  MjdEnd = Mjd(year,month,day);

  // Запрос о разности между эфемеридным и Всемирным временем
  ETminUT ( (0.5*(MjdStart+MjdEnd)-MJD_J2000)/36525.0, ET_UT, valid );

  if ( valid ) {
    cout << " Difference ET-UT (proposal " << fixed << setw(6)
         << setprecision (1) << ET_UT << " sec)        ... ";
    cin >> ET_UT; cin.ignore(81,'\n');
  }
  else {
    cout << " Difference ET-UT (sec)" << setw(27) << right << "... ";
    cin >> ET_UT; cin.ignore(81,'\n');
  }

  // Запрос о географических координатах
  cout << " Observer's coordinates: East longitude [deg]      ";
  cin >> Lambda; cin.ignore(81,'\n');
  cout << "                         latitude       [deg]  .. ";
  cin >> Phi; cin.ignore(81,'\n');

  Lambda *= Rad; Phi *= Rad;
  R_Obs = Site (Lambda, Phi);
}


//-------------------------------------------------------------------------------
//
// FG: Вычисление координат в главной плоскости
//   T_ET      Время в юлианских столетиях после J2000
//   ET_UT     Разность между эфемеридным и Всемирным временем, с
//   e         Направление на звезду (экваториальные координаты)
//   R_Obs     Геоцентрические координаты наблюдателя, км
//   f, g      Координаты в главной плоскости, км
//   s         Квадрат расстояния между осью тени и наблюдателем, км
//   ChebMoonEqu координаты Луны (приближение Чебышева)
//
//-------------------------------------------------------------------------------
void FG ( double T_ET, double ET_UT, const Vec3D& e, const Vec3D& R_Obs,
          double& f, double& g, double& s )
{
  //
  // Переменные
  //
```

```
  double  MjdUT;
  Vec3D   e_x, e_y, r_Moon, r_Obs;

  // Единичные векторы в главной плоскости
  e_x = Cross ( Vec3D(0,0,1), e );
  e_x = e_x / Norm(e_x);
  e_y = Cross ( e, e_x );

  // Координаты Луны
  r_Moon = ChebMoonEqu.Value(T_ET);

  // Координаты наблюдателя
  MjdUT = (36525.0*T_ET+MJD_J2000) - ET_UT/86400;
  r_Obs = R_z(-GMST(MjdUT)) * R_Obs;

  // Координаты в главной плоскости
  f = Dot ( e_x, r_Moon - r_Obs );
  g = Dot ( e_y, r_Moon - r_Obs );
  s = f*f + g*g - R_Moon*R_Moon;
}

//------------------------------------------------------------------------------
// Contact: Вычисление позиционного угла, дифференциальных коэффициентов и определение
//          видимости (в связи в высотой и яркостью неба)
//   T_ET       Время в юлианских столетиях после J2000
//   ET_UT      Разность между эфемеридным и Всемирным временем, с
//   e          Направление на звезду (экваториальные координаты)
//   R_Obs      Геоцентрические координаты наблюдателя, км
//   InOut      Указатель покрытия или открытия
//   mag        Звездная величина, m
//   PosAngle   Позиционный угол, рад
//   a          Коэффициент по долготе, мин./'
//   b          Коэффициент по долготе, мин./'
//   Visible    Указатель видимости покрытия
//------------------------------------------------------------------------------
void Contact ( double T_ET, double ET_UT, const Vec3D& e, const Vec3D& R_Obs,
               enEvent InOut, double mag,
               double& PosAngle, double& a, double& b, bool& Visible )
{
  //
  // Константы
  //
  const double cy = 36525.0*1440.0;   // Минуты за столетие
  const double dt = 1.0;              // Шаг по времени, мин.

  //
  // Переменные
  //
  double  MjdUT;
  Vec3D   e_x, e_y, r_Moon, r_Obs, e_Obs, r_Sun;
  Vec3D   dt_dr, dr_dlambda, dr_dphi;
  double  s_0, f, g, ff, gg, fdot, gdot;
  double  Elev, ElevSun, PosAngSun, DPosAng, i;
  bool    BrightLimb;

  // Единичные векторы в главной плоскости
  e_x = Cross ( Vec3D(0,0,1), e );
  e_x = e_x / Norm(e_x);
  e_y = Cross ( e, e_x );

  // Координаты Луны
```

```
r_Moon = ChebMoonEqu.Value(T_ET);

// Геоцентрические координаты наблюдателя
MjdUT = (36525.0*T_ET+MJD_J2000) - ET_UT/86400;

r_Obs = R_z(-GMST(MjdUT)) * R_Obs;
e_Obs = r_Obs / Norm(r_Obs);

// Высота звезды над горизонтом
Elev = asin ( Dot(e_Obs,e) );

// Координаты в главной плоскости
f = Dot ( e_x, r_Moon - r_Obs );
g = Dot ( e_y, r_Moon - r_Obs );

// Позиционный угол
PosAngle = Modulo(atan2(-f, -g), pi2);

// Производные по времени координат в главной плоскости
FG ( T_ET+dt/cy, ET_UT, e, R_Obs, ff, gg, s_0 );
fdot = (ff-f)/dt;  // [km/min]
gdot = (gg-g)/dt;

// Производные геоцентрических координат наблюдателя
dr_dlambda = Cross ( Vec3D(0,0,1), r_Obs );
dr_dphi    = Cross ( r_Obs, dr_dlambda / Norm(dr_dlambda) );

// Вычисление дифференциальных коэффициентов
dt_dr = (f*e_x+g*e_y) / (f*fdot+g*gdot);

a = Dot ( dt_dr, dr_dlambda );
b = Dot ( dt_dr, dr_dphi    );

// Вычисление координат и высоты Солнца, позиционного угла
r_Sun = AU*SunEqu(T_ET);
ElevSun = asin ( Dot ( e_Obs, r_Sun/Norm(r_Sun) ) );
PosAngSun = PosAng(r_Moon-r_Obs, r_Sun-r_Moon);
DPosAng = Modulo(PosAngle-PosAngSun+pi, pi2) - pi;
BrightLimb = ( fabs(DPosAng) < pi/2.0 );

// Угол фазы Луны
i = acos ( Dot(-r_Moon, r_Sun-r_Moon) / (Norm(r_Sun-r_Moon)*Norm(r_Moon)) );

// Условия видимости
Visible = ( // Минимальная высота звезды
            ( Elev > +5.0*Rad )                   ||
            ( Elev > +2.0*Rad ) && ( mag <= 1.9 )
          ) &&
          ( // Минимальное погружение Солнца под горизонт
            ( ElevSun < -6.0*Rad ) && ( mag <= 7.5 ) ||
            ( ElevSun < -3.0*Rad ) && ( mag <= 5.5 ) ||
            ( ElevSun < -0.5*Rad ) && ( mag <= 4.5 ) ||
                                      ( mag <= 1.9 )
          ) &&
          ( // Фаза Луны
            ( i > 36.0*Rad )                   ||
            ( i > 24.0*Rad ) && ( mag <= 6.5 ) ||
            ( i > 12.0*Rad ) && ( mag <= 5.5 ) ||
                                ( mag <= 3.5 )
          ) &&
          ( // Освещенность лимба
            ( !BrightLimb ) && (InOut==in )                ||
            ( !BrightLimb ) && (InOut==out) && (mag<=6.5) ||
```

```
            (  BrightLimb ) && (InOut==in ) && (mag<=4 5) ||
            (  BrightLimb ) && (InOut==out) && (mag<=3 5)
          ).
}
//----------------------------------------------------------------------------
// Examine· Исследование возможности покрытия
//   T1       Начало исследуемого промежутка
//   T2       Конец исследуемого промежутка
//   RA1      Прямое восхождение Луны в момент T1. рад
//   RA2      Прямое восхождение Луны в момент T2. рад
//   ET_UT    Разница между эфемеридным и Всемирным временем. с
//   R_Obs    Геоцетрические координаты наблюдателя. км
//   aStar    Исследуемая звезда
//   In       Событие покрытия
//   Out      Событие открытия
//----------------------------------------------------------------------------
void Examine ( double T1. double T2. double RA1. double RA2. double ET_UT.
               const Vec3D& R_Obs. Star& aStar. Event& In. Event& Out )
{
const double cy = 876600 0.      // Часы за столетие
  const double dT = 0 25/cy:     // Величина шага
  const double DT = 2.25/cy:     // Исследуемый интервал [-DT-dT.+DT+dT]
Vec3D   e.
  bool    Conj. Visible:
  int     n_found. n_root. i:
  double  T_Conj. T. T_Cont[2]. MjdUT:
  double  s_minus. s_0. s_plus. xe. ye. root[2]:
  double  f. g. PosAng. a. b:

  In = Out = Event(): // Default values

  // Видимое положение звезды
  e = aStar.Apparent():

  // Определение момента соединения
  Conjunct ( T1. T2. RA1. RA2. e. Conj. T_Conj ):
  if (!Conj) return.

  // Определение моментов контакта
  n_found = 0.
  T = T_Conj - DT:
  FG (T-dT. ET_UT. e. R_Obs. f. g. s_minus).

  while (true) {

    // Интерполяция расстояния в главной плоскости и нахождение моментов контакта
    FG (T    . ET_UT. e. R_Obs. f. g. s_0   ):
    FG (T+dT. ET_UT. e. R_Obs. f. g. s_plus):
    Quad (s_minus. s_0. s_plus. xe. ye. root[0]. root[1]. n_root):
    for (i=0.i<n_root:i++)
      T_Cont[n_found+i] = T + root[i]*dT:
    n_found += n_root:
    Conj = (n_found==2).
    if ( (Conj) || (T_Conj+DT<T) ) break: // Exit loop
    T+=2 0*dT: // Приращение по времени
    s_minus = s_plus:
  }:

  // Контакты (покрытие. открытие)
  if (Conj) {
```

```
    for (i=in;i<=out;i++)  {
      MjdUT = 36525.0*T_Cont[i] + MJD_J2000 - ET_UT/86400.0;
      Contact ( T_Cont[i]. ET_UT. e. R_Obs. enEvent(i). aStar.Mag().
                PosAng. a. b. Visible );
      if (i==in)  In  = Event(aStar. MjdUT. in . Visible. PosAng. a. b).
      if (i==out) Out = Event(aStar. MjdUT. out. Visible. PosAng. a. b).
    }
  }
}

//------------------------------------------------------------------------------
//
// Основная программа
//
//------------------------------------------------------------------------------
void main(int argc. char* argv[])
{
  //
  // Постоянные
  //
  const double Margin    = 3.0 / (36525.0*24 0):  // 3h [cy]

  //
  // Переменные
  //
  Star*     Stars = NULL:     // Звездный каталог
  int       NStars. iStar:
  double    MjdStart. MjdEnd. ET_UT. T. TEnd:
  double    RA_min. RA_max:
  Vec3D     R_Obs.
  Event     In. Out.
  char      InputFile[APC_MaxFilename] = "":
  char      OutputFile[APC_MaxFilename] = "":
  bool      FoundInputfile  = false:
  bool      FoundOutputfile = false:
  ofstream  OutFile.

  // Title
  cout << endl
       << "          OCCULT: покрытия звезд Луной " << endl
       << "         (c) 1999 Oliver Montenbruck. Thomas Pfleger " << endl
       << endl:

  // Поиск каталожного файла и (дополнительного) имени файла вывода
  GetFilenames( argc. argv. "Occult dat". InputFile. FoundInputfile.
                OutputFile. FoundOutputfile ):

  // Завершение программы. если файл ввода не найден
  if (!FoundInputfile) {
    cerr << "Terminating program " << endl:
    exit(-1).
  }

  ReadCatalogue (InputFile. Stars. NStars):

  // Завершение программы. если звездный каталог не может быть прочитан
  if (NStars == 0) {
    cerr << "Terminating program." << endl:
    exit(-1):
  }
```

```
  // Ввод данных пользователя (исследуемый промежуток, положение наблюдателя и т. д.)
  GetInput (MjdStart, MjdEnd, ET_UT, R_Obs);

  // Перенаправление вывода в файл
  if (FoundOutputfile) {
    OutFile.open(OutputFile);
    if (OutFile.is_open())
      cout = OutFile;
  }

  // Печать заголовка
  cout
   << endl
   << "    Date         Name       m_v   D/R       UT         Pos       a       b   "
   << endl
   << "                            mag                        deg     m/deg   m/deg"
   << endl;

  // Поиск покрытий в течение последовательных промежутков времени (эфемеридное время)
  T    = (MjdStart-MJD_J2000)/36525.0;
  TEnd = (MjdEnd  -MJD_J2000)/36525.0;

  do {

    // Приближение Чебышева для лунных координат
    ChebMoonEqu.Fit(T-Margin, T+Interval+Margin);

    RA_min = ChebMoonEqu.Value(T)[phi];
    RA_max = ChebMoonEqu.Value(T+Interval)[phi];

    // Эпоха для вычисления видимых положений
    Star::SetEpoch(T+Interval/2.0);

    // Цикл для звездного каталога
    for (iStar=0; iStar<NStars; iStar++) {

      // Исследование возможности покрытий
      Examine ( T, T+Interval, RA_min, RA_max, ET_UT, R_Obs, Stars[iStar], In, Out );

      // Печать: покрытие или открытие, если оно наблюдаемо
      if (In.IsVisible() ) cout << In;
      if (Out.IsVisible()) cout << Out;
    }

    T += Interval;  // Следующий промежуток
  }
  while (T<TEnd);

  if (OutFile.is_open()) OutFile.close();
  if (Stars!=NULL) delete[] Stars;
}
```

В качестве примера приведем два ряда вычислений покрытий звезд Плеяд, происходивших в 1989 году, произведенных с помощью Occult. Выберем период между 15 сентября и 15 ноября этого года. Вычисления производятся для Мюнхена, имеющего координаты 48°,1 северной широты и 11°,6 восточной долготы. Примем разницу между эфемеридным и Всемирным временем равной 57 секундам в соответствии с данными табл. 9.2. В нашем примере данные, введенные пользователем, выделены курсивом.

```
        OCCULT: occultations of stars by the Moon
       (c) 1999 Oliver Montenbruck, Thomas Pfleger

 Using default input file Occult.dat

 9 stars read from catalogue

 Period of time for prediction of occultations
  first date (yyyy mm dd)                      ... 1989 09 15
  last  date (yyyy mm dd)                      ... 1989 11 15
 Difference ET-UT (proposal:  57.0 sec)        ... 57.0
 Observer's coordinates: East longitude [deg] ... 11.60
                          latitude      [deg] ... 48.10

    Date        Name       m_v   D/R     UT       Pos      a      b
                           mag                    deg    m/deg  m/deg
 1989/09/19  Celaeno       5.4    R   22·41:00    222     0.0    2.1
 1989/09/19  Taygeta       4.4    D   22:01:57     67     0.1    1.7
 1989/09/19  Taygeta       4.4    R   23:00:37    251     0.5    1.7
 1989/09/19  Maia          4.0    D   22:15.34     98     0.5    1.3
 1989/09/19  Maia          4.0    R   23:07:32    220     0.1    2.2
 1989/09/19  Asterope      5.9    R   23:20:36    256     0.6    1.6
 1989/11/13  Alcyone       3.0    D   19:00:10    122     1.1    0.6
 1989/11/13  Alcyone       3.0    R   19:36:07    198    -0.3    2.9
```

Мы обнаруживаем, что в течение двух месяцев произошло пять покрытий звезд в Плеядах, причем покрытия или открытия удовлетворяют принятым критериям и доступны наблюдениям.

Чтобы сократить время вычислений и объем выходных данных, файл ввода в нашем примере содержит лишь небольшой список звезд. Прилагаемый к книге компакт-диск содержит дополнительный файл **ZC.dat**, в котором приведены данные о 3546 звездах из каталога зодиакальных звезд. Координаты заимствованы из каталога PPM, обеспечивающего значительно большую точность, нежели исходный каталог зодиакальных звезд. Для использования данных **ZC.dat** или иного специального звездного каталога при запуске программы `Occult` в командной строке после названия программы вводится имя соответствующего файла (см. раздел П.1.3).

10.6. Оценка из наблюдений величины ΔT=ET–UT

Наблюдения покрытий позволяют без особого труда определить разницу между эфемеридным и Всемирным временем и проследить, таким образом, замедление вращения Земли.

Для этого сравним полученный из наблюдений момент покрытия или открытия со значением $t_{\mathrm{ET-UT}}$, предсказанным в предположении, что ΔT = ET–UT = 0. Если наблюдаемый момент контакта t_{UT} определен по Всемирному времени, его значение по эфемеридному времени будет

$$t_{\mathrm{ET}} = t_{\mathrm{UT}} + \Delta T.$$

Значение t_{ET} по эфемеридному времени, однако, не совпадает с вычисленным значением t_0, потому что рассчитанное в предположении ET = UT звездное время по Гринвичу отличается от действительного на величину $\Delta\Theta_0 \approx \Delta T$. Следовательно, покрытие фактически было рассчитано для пункта наблюдений, находящегося на величину $\Delta\lambda = 15°/\mathrm{h}\cdot\Delta T$ восточнее реального пункта наблюдений. Эта ошибка может быть легко исправлена с учетом дифференциального коэффициента a

$$t_{ET} = t_0 - a(0\overset{\circ}{.}25/\mathrm{m}\cdot\Delta T).$$

После подстановки и преобразования получается соотношение

$$\Delta T = \frac{t_0 - t_{UT}}{1 + a\cdot 0\overset{\circ}{.}25/\mathrm{m}},$$

из которого определяется величина ΔT.

В таблице 10.2 приведены результаты наблюдений пяти покрытий, наблюдавшихся между 1896 и 1990 годами в различных пунктах Европы. Легко заметить, что разница между эфемеридным и Всемирным временем за прошедшее столетие увеличилась примерно на минуту, что наглядно демонстрирует замедление вращения Земли. Значение ΔT определяется из этих наблюдений с погрешностью 1–5 секунд, что соответствует точности предвычислений, произведенных с помощью программы `Occult`. Эти погрешности вызваны как небольшими ошибками в определении звездных и лунных координат, так и неровностями лунного лимба.

Таблица 10.2. Примеры определения разницы между эфемеридным и Всемирным временем по наблюдениям покрытий звезд

Звезда	Пункт наблюдения		Время наблюдения		Вычисленное время в предположении ET=UT	a	ΔT, с
	λ	φ	Дата	UT			
17 Tau	Королевск. обс., Берлин +13°.396	+52°.505	26.09.1896	20:32:10	20:32:02	$-0\overset{m}{.}7/°$	−10
57B Sco	Университет Варшавы +21°.030	+52°.218	24.01.1930	05:30:45	05:31:08	$-0\overset{m}{.}3/°$	25
μ Cet	Обсерватория Штутгарта +9°.197	+48°.784	10.02.1962	21:12:45	21:13:22	$+0\overset{m}{.}6/°$	32
δ Psc	Обсерватория Штутгарта +9°.197	+48°.784	13.01.1970	19:35:12	19:36:03	$+1\overset{m}{.}1/°$	40
ε Ari	Св. Августин +7°.177	+50°.775	30.11.1990	22:01:32	22:02:59	$+1\overset{m}{.}9/°$	59

11. Вычисление орбит

Классическая задача вычисления орбиты планеты, кометы или астероида заключается в нахождении ее элементов из наименьшего числа измеренных координат. По существу, это задача обратная вычислению эфемерид, когда координаты светила находятся по известным элементам орбиты. Каждое наблюдение с земной поверхности дает две сферические координаты для некоторого момента времени. Мы можем выбрать экваториальную систему координат (прямое восхождение и склонение) или эклиптическую (эклиптические широта и долгота). Поскольку расстояние до светила при таких наблюдениях остается неизвестным, оно не может быть использовано при вычислении орбиты. Для нашей задачи достаточно трех наблюдений, поскольку для определения шести элементов орбиты требуется такое же число независимых величин.

Описываемый здесь метод вычисления орбит восходит к К. Ф. Гауссу, частично он был пересмотрен и усовершенствован Буцериусом. Существо процедуры состоит, с одной стороны, в определении элементов орбиты по двум позиционным векторам и, с другой стороны, в итерации так называемых соотношений площадей треугольников, отражающих геометрическое расположение трех наблюденных точек в плоскости орбиты. Процедура дополняется решением уравнения Гаусса—Лагранжа, дающего надежный критерий для выбора между различными решениями задачи вычисления орбиты.

11.1. Вычисление орбиты по двум позиционным векторам

Элементы орбиты обычно используются для описания орбиты планеты или кометы. Для вычисления орбиты, впрочем, более удобным является другой метод ее описания. Как известно, орбита может быть определена по скорости и положению небесного тела в определенный момент или по двум его положениям на орбите в соответствующие моменты времени. Первый способ используется в методе вычисления орбит Лапласа, метод же Гаусса основан на использовании двух позиционных векторов. В этом разделе мы будем иметь дело с вычислением элементов орбиты на основании двух положений тела $\boldsymbol{r}_a$ и $\boldsymbol{r}_b$ в моменты времени t_a и t_b. Задача вычисления орбиты, таким образом, сводится к определению двух гелиоцентрических положений из трех наблюдаемых направлений на объект.

Вначале обсудим промежуточное действие — нахождение *отношения площадей сектора и треугольника*, представляющее наиболее трудный этап в процессе

вычисления орбиты. Эта величина, как будет показано в следующих разделах, особенно важна для дальнейших этапов вычислений.

11.1.1. Отношение площадей сектора и треугольника

Площадь Δ треугольника, определяемого векторами $\boldsymbol{r}_a$ и $\boldsymbol{r}_b$ (рис. 11.1), зависит от длин сторон r_a и r_b и образованного ими угла $\nu_b - \nu_a$, который, однако, должен быть менее 180°. Следовательно:

$$\Delta = \frac{1}{2} r_a r_b \cdot \sin(\nu_b - \nu_a). \tag{11.1}$$

Здесь ν_a и ν_b — значения истинной аномалии в конечных пунктах рассматриваемого участка орбиты.

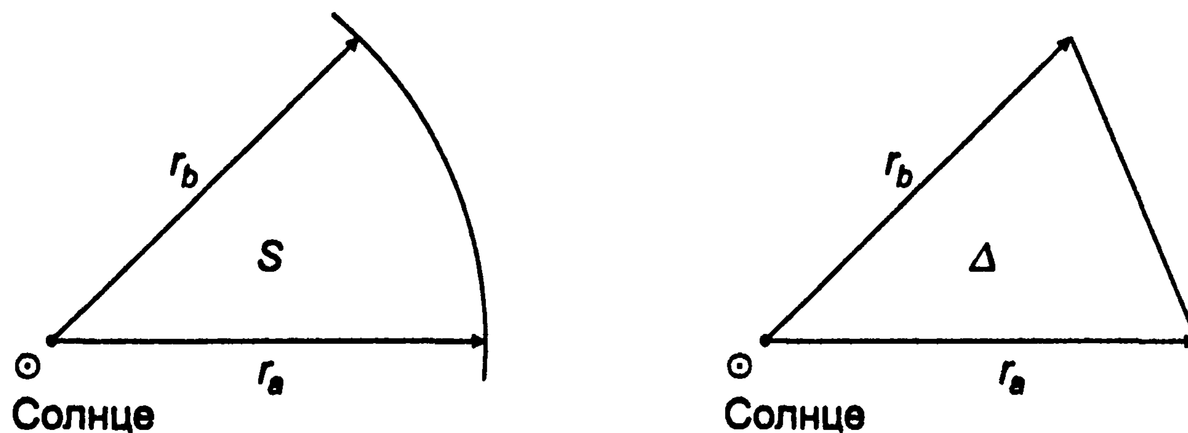


Рис. 11.1. Площади сектора и треугольника

Площадь S сектора, ограниченного $\boldsymbol{r}_a$ и $\boldsymbol{r}_b$ и дугой орбиты, заключенной между ними, в силу второго закона Кеплера (закон равных площадей) пропорциональна разнице между моментами времени t_a и t_b:

$$S = \frac{1}{2}\sqrt{GM_\odot} \cdot \sqrt{a(1-e^2)} \cdot (t_b - t_a). \tag{11.2}$$

Здесь a и e обозначают соответственно большую полуось и эксцентриситет орбиты, соединяющей данные точки (см. главу 4). Если мы подставим *фокальный параметр* (semi-latus rectum — половина хорды, проведенной через фокус параллельно малой оси) $p = a(1 - e^2)$, получим выражение

$$\eta = \frac{S}{\Delta} = \frac{\sqrt{p} \cdot \tau}{r_a r_b \cdot \sin(\nu_b - \nu_a)}, \tag{11.3}$$

для отношения η двух площадей, где для простоты вводится обозначение

$$\tau = \sqrt{GM_\odot} \cdot (t_b - t_a). \tag{11.4}$$

Как будет видно, уравнение для η содержит фокальный параметр p, который не был прежде выражен через $\boldsymbol{r}_a$ и $\boldsymbol{r}_b$. Если мы попытаемся исключить фокальный параметр, используя известные уравнения задачи двух тел, то обнаружим, что

невозможно получить для η алгебраическое уравнение, имеющее решение. Вместо этого мы получим трансцендентное уравнение[1]

$$\eta = 1 + \frac{m}{\eta^2} \cdot W\left(\frac{m}{\eta^2} - l\right), \qquad (11.5)$$

позволяющее определить η и содержащее вспомогательные переменные

$$\begin{aligned} m &= \frac{\tau^2}{\sqrt{2(r_a r_b + \boldsymbol{r}_a \cdot \boldsymbol{r}_b)^3}}, \\ l &= \frac{r_a + r_b}{2\sqrt{2(r_a r_b + \boldsymbol{r}_a \cdot \boldsymbol{r}_b)}} - \frac{1}{2}. \end{aligned} \qquad (11.6)$$

Здесь функция W задается следующим образом:

$$W(w) = \begin{cases} \dfrac{2g - \sin(2g)}{\sin^3(g)}, & g = 2\arcsin\sqrt{w} \quad 0 < w < 1 \\ \dfrac{4}{3} + \dfrac{4 \cdot 6}{3 \cdot 5} w + \dfrac{4 \cdot 6 \cdot 8}{3 \cdot 5 \cdot 7} w^2 + \ldots & w \approx 0 \\ \dfrac{\operatorname{sh}(2g) - 2g}{\operatorname{sh}^3(g)}, & g = 2\operatorname{arsh}\sqrt{-w} \quad w < 0 \end{cases} \qquad (11.7)$$

Если мы запишем

$$f(x) = 1 - x + \frac{m}{x^2} \cdot W\left(\frac{m}{x^2} - l\right),$$

тогда искомое значение η соответствует точке, в которой функция f обращается в нуль.

```
// F = 1 - eta +(m/eta**2)*W(m/eta**2-l)
double F (double eta, double m, double l)
const double eps = 100.0 * eps_mach;
double  w,W,a,n,g;
w = m/(eta*eta)-l;
if (fabs(w)<0.1) { // Разложение в ряд
      W = a = 4.0/3.0; n = 0.0;
      do {
        n += 1.0;  a *= w*(n+2.0)/(n+1.5);  W += a;
      }
      while (fabs(a) >= eps);
    }
    else {
      if (w > 0.0) {
        g = 2.0*asin(sqrt(w));
        W = (2.0*g - sin(2.0*g)) / pow(sin(g), 3);
```

[1] Вывод этого уравнения занимает несколько страниц, поэтому здесь он не приводится. Читатель может найти интересующие его подробности в соответствующих работах, включенных в библиографию.

```
    }
    else {
      g = 2.0*log(sqrt(-w)+sqrt(1.0-w)); // =2.0*arsinh(sqrt(-w))
      W = (sinh(2 0*g) - 2.0*g) / pow(sinh(g), 3);
    }
  }

  return ( 1.0 - eta + (w+1)*W );
}
```

Поскольку стандартная библиотека C++ не поддерживает обратные гиперболические функции, необходимо выразить функцию $\operatorname{arsh} x$ через натуральный логарифм: $\operatorname{arsh} x = \ln(x + \sqrt{1 + x^2})$.

Для определения значения η методом итерации удобнее всего использовать метод секущих. Имея два приближения η_{i-1} и η_i, мы получим уточненное значение η_{i+1} с помощью выражения

$$\eta_{i+1} = \eta_i - f(\eta_i) \cdot \frac{\eta_i - \eta_{i-1}}{f(\eta_i) - f(\eta_{i-1})}.$$

Геометрически это соответствует нуль-точке секущей, проходящей через точки $(\eta_{i-1}, f(\eta_{i-1}))$ и $(\eta_i, f(\eta_i))$ на кривой, заданной f. При последовательном выполнении этого шага итерация приводит к искомому значению отношения площадей сектора и треугольника. Подходящие начальные значения

$$\eta_1 = \eta_{Hansen} + 0{,}1 \qquad \text{и} \qquad \eta_2 = \eta_{Hansen}$$

дает известное приближение Хансена

$$\eta_{Hansen} = \frac{12}{22} + \frac{10}{22}\sqrt{1 + \frac{44}{9}\frac{m}{l + 5/6}}. \tag{11.8}$$

Таким образом, можно составить следующую функцию для вычисления отношения площадей сектора и треугольника.

```
//------------------------------------------------------------------------------
// FindEta: Вычисление отношения площадей сектора и треугольника для двух заданных положений
//          и промежуток времени между ними
//   r_a      Положение в первый момент, а. е.
//   r_a      Положение во второй момент, а  е.
//   tau      Промежуток времени между положениями r_a и r_b (kGauss * dT, сут.)
// Возвращаемые значения.  Отношение площадей сектора и треугольника
//------------------------------------------------------------------------------
double FindEta (const Vec3D& r_a, const Vec3D& r_b, double tau)
{
   const int maxit = 30;
  const double delta = 100.0*eps_mach;
  int    i;
  double kappa, m, l, s_a, s_b, eta_min, eta1, eta2, F1, F2, d_eta,
  s_a = Norm(r_a);
  s_b = Norm(r_b);
  kappa = sqrt ( 2.0*(s_a*s_b+Dot(r_a,r_b)) ),
```

```
  m = tau*tau / pow(kappa,3);
  l = (s_a+s_b) / (2.0*kappa) - 0.5;
  eta_min = sqrt(m/(l+1 0));
  // Приближение Хансена
  eta2 = ( 12.0 + 10.0*sqrt(1.0+(44.0/9.0)*m /(l+5.0/6 0)) ) / 22.0.
  eta1 = eta2 + 0.1.
  // Метод секущих
  F1 = F(eta1, m, l);
  F2 = F(eta2, m, l).
  i = 0;
  while (fabs(F2-F1) > delta)
  {
    d_eta = -F2*(eta2-eta1)/(F2-F1).
    eta1 = eta2; F1 = F2;
    while (eta2+d_eta<=eta_min)  d_eta *= 0.5;
    eta2 += d_eta;
    F2 = F(eta2,m,l); ++i;
    if ( i == maxit ) {
      cerr << " Convergence problems in FindEta" << endl.
      break;
    }
  }

  return eta2;
}
```

11.1.2. Элементы орбиты

Орбита небесного тела, проходящая через точки r_a и r_b, всегда лежит в плоскости, заданной этими точками и Солнцем[1]. Для определения наклонения i этой плоскости к эклиптике, сначала получим единичные векторы $\boldsymbol{e}_a$ и $\boldsymbol{e}_0$, лежащие в орбитальной плоскости:

$$\boldsymbol{e}_a = \frac{\boldsymbol{r}_a}{|\boldsymbol{r}_a|} \tag{11.9}$$

$$\boldsymbol{e}_0 = \frac{\boldsymbol{r}_0}{|\boldsymbol{r}_0|}, \quad \text{где} \quad \boldsymbol{r}_0 = \boldsymbol{r}_b - (\boldsymbol{r}_b \cdot \boldsymbol{e}_a)\boldsymbol{e}_a. \tag{11.10}$$

Эти векторы показаны на рис. 11.2: $\boldsymbol{e}_a$ направлен вдоль вектора $\boldsymbol{r}_a$, а векторы $\boldsymbol{r}_0$ и $\boldsymbol{e}_0$ ему перпендикулярны. Произведение $\boldsymbol{e}_a$ и $\boldsymbol{e}_0$ дает единичный Гауссов вектор $\boldsymbol{R}$, перпендикулярный плоскости орбиты ($|\boldsymbol{R}| = 1$):

$$\boldsymbol{R} = \boldsymbol{e}_a \times \boldsymbol{e}_0, \quad \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = \begin{pmatrix} y_a z_0 - z_a y_0 \\ z_a x_0 - x_a z_0 \\ x_a y_0 - y_a x_0 \end{pmatrix}. \tag{11.11}$$

---

[1] Вообще говоря, центром масс Солнечной системы, который не совпадает с центром Солнца. — *Примеч. ред.*

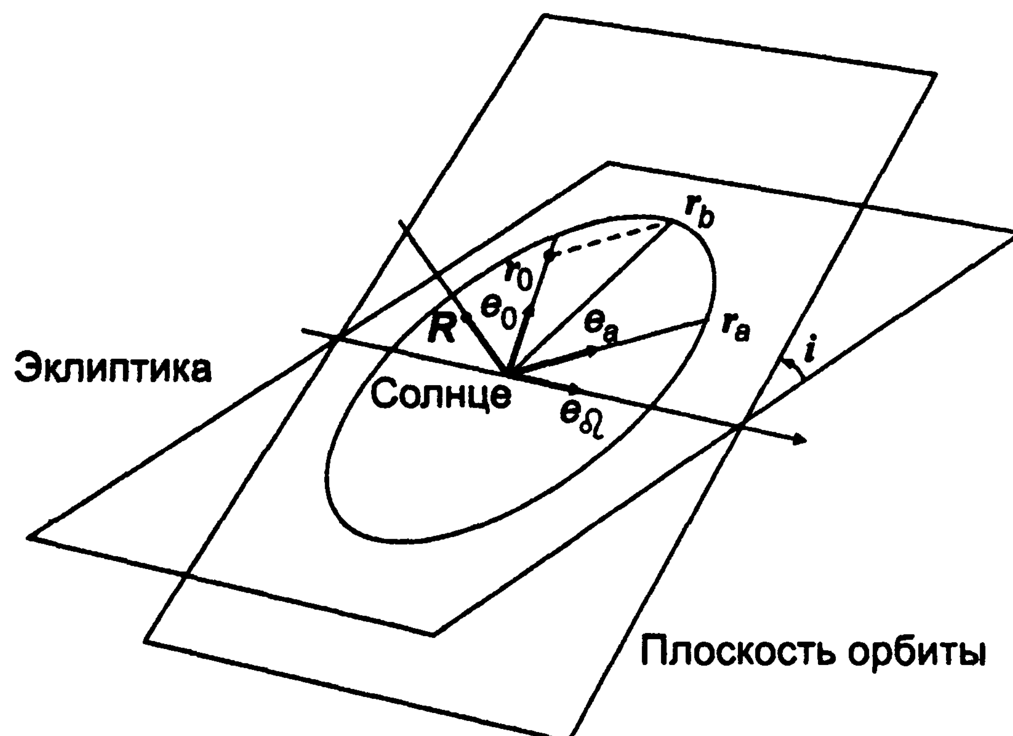


Рис. 11.2. Вспомогательные переменные, используемые для определения плоскости орбиты

Вектор $\boldsymbol{R}$ направлен к точке с координатами $l = ☊ - 90°$ по эклиптической долготе и $b = 90° - i$ по эклиптической широте, поэтому он может быть выражен через элементы ☊ и i:

$$\boldsymbol{R} = \begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = \begin{pmatrix} +\cos(90° - i)\cos(☊ - 90°) \\ +\cos(90° - i)\sin(☊ - 90°) \\ +\sin(90° - i) \end{pmatrix} = \begin{pmatrix} +\sin i \sin ☊ \\ -\sin i \cos ☊ \\ +\cos i \end{pmatrix}. \quad (11.12)$$

Таким образом, получаем три уравнения для долготы узла и наклонения орбиты, имеющие единственное решение:

$$☊ = 90° + \operatorname{arctg}(R_y / R_x) = \operatorname{arctg}(-R_y / R_x), \quad (11.13)$$

$$i = 90° - \arcsin(R_z). \quad (11.14)$$

Оба угла относятся к тому же равноденствию, что и векторы $\boldsymbol{r}_a$ и $\boldsymbol{r}_b$. Положение линии узлов позволяет теперь определить аргумент широты u_a, соответствующий углу между позиционным вектором $\boldsymbol{r}_a$ и направлением на восходящий узел орбиты. Для этого угла имеем

$$\cos u_a = \boldsymbol{e}_a \cdot \boldsymbol{r}_☊ = x_a \cdot \cos ☊ + y_a \cdot \sin ☊,$$

$$\cos(u_a + 90°) = \boldsymbol{e}_0 \cdot \boldsymbol{e}_☊ = x_0 \cdot \cos ☊ + y_0 \cdot \sin ☊,$$

где

$$\boldsymbol{e}_☊ = \begin{pmatrix} \cos ☊ \\ \sin ☊ \\ 0 \end{pmatrix}$$

единичный вектор, направленный вдоль линии узлов. Таким образом

$$u_a = \operatorname{arctg}\left(\frac{-x_0 \cdot \cos ☊ - y_0 \cdot \sin ☊}{+x_a \cdot \cos ☊ + y_a \cdot \sin ☊}\right) = \operatorname{arctg}\left(\frac{+x_0 \cdot R_y - y_0 \cdot R_x}{-x_a \cdot R_y + y_a \cdot R_x}\right). \quad (11.15)$$

Для определения оставшихся элементов орбиты нам требуется вычислить отношение площадей сектора и треугольника, как это было описано в предыдущем разделе. После этого можем выразить фокальный параметр

$$p = \left(\frac{2 \cdot \Delta \cdot \eta}{\tau}\right)^2$$

через площадь треугольника, заданного векторами $\boldsymbol{r}_a$ и $\boldsymbol{r}_b$

$$\Delta = \frac{1}{2} r_a r_b \cdot \sin(\nu_b - \nu_a) = \frac{1}{2} r_a r_0,$$

и интервал τ.

Форма орбиты определяется ее эксцентриситетом e, который может быть найден из уравнения конического сечения

$$r = \frac{p}{1 + e \cdot \cos\nu}.$$

Решение для $e\cos\nu$ дает

$$e \cdot \cos\nu_a = p / r_a - 1,$$
$$e \cdot \cos\nu_b = p / r_b - 1.$$

Приняв во внимание то обстоятельство, что

$$\cos\nu_b = \cos\nu_a \cos(\nu_b - \nu_a) - \sin\nu_a \sin(\nu_b - \nu_a) =$$
$$= \cos\nu_a \cdot \left(\frac{\boldsymbol{r}_b \cdot \boldsymbol{e}_a}{r_b}\right) - \sin\nu_a \cdot \left(\frac{r_0}{r_b}\right),$$

путем подстановки мы получим два уравнения

$$e \cdot \cos\nu_a = p / r_a - 1,$$
$$e \cdot \sin\nu_a = \left\{ (p/r_a - 1)\left(\frac{\boldsymbol{r}_b \cdot \boldsymbol{e}_a}{r_b}\right) - (p/r_b - 1) \right\} \Big/ \left(\frac{r_0}{r_b}\right),$$

из которых могут быть получены значения эксцентриситета и истинной аномалии в момент времени t_a:

$$e = \sqrt{(e \cdot \cos(\nu_a))^2 + (e \cdot \sin(\nu_a))^2},$$
$$\nu_a = \operatorname{arctg}\left(\frac{e \cdot \sin(\nu_a)}{e \cdot \cos(\nu_a)}\right).$$

Аргумент и долгота перигелия получаются из разницы между аргументом долготы и истинной аномалией:

$$\omega = u_a - \nu_a \tag{11.16}$$

$$\varpi = u_a - \nu_a + \Omega. \tag{11.17}$$

Знание эксцентриситета позволяет нам определить форму орбиты, заключенной между r_a и r_b: это или эллипс ($e < 1$), или гипербола ($e > 1$). Параболическая орбита здесь не будет рассматриваться, поскольку на практике трудно ожидать, что e будет в точности равно единице. Из значений фокального параметра и эксцентриситета мы можем также получить величины малой полуоси и расстояния перигелия:

$$a = \frac{p}{1 - e^2}, \tag{11.18}$$

$$q = \frac{p}{1 + e}. \tag{11.19}$$

Необходимо отметить, что по определению величина малой полуоси гиперболической орбиты имеет отрицательное значение.

Теперь нам известны все элементы орбиты, определяющие ее форму (e), размеры (a) и ориентацию в пространстве (i, ☊, ω). Шестой и последний элемент, который должен быть вычислен, — это время прохождения через перигелий, то есть время, когда тело проходит ближайшую к Солнцу точку орбиты. В случае эллиптической орбиты сперва необходимо определить значение эксцентрической аномалии E_a с помощью следующих уравнений:

$$\cos E_a = \frac{\cos \nu_a + e}{1 + e \cdot \cos \nu_a},$$

$$\sin E_a = \frac{\sqrt{1 - e^2} \sin \nu_a}{1 + e \cdot \cos \nu_a}.$$

Эти уравнения получаются в результате исключения радиуса из формулы (4.5) благодаря использованию уравнения конического сечения. Величина средней аномалии M, связанная с E, определяется из уравнения Кеплера

$$M_a = E_a - e \cdot \sin E_a \quad \text{радиан}.$$

Период обращения определяется из третьего закона Кеплера

$$T = 2\pi \cdot \sqrt{\frac{a^3}{GM_\odot}},$$

средняя аномалия изменяется за сутки на

$$n = 2\pi \cdot \frac{1^d}{T} = \sqrt{\frac{GM_\odot}{a^3}} \cdot 1^d,$$

поэтому время прохождения перигелия

$$t_0 = t_a - M_a \Big/ \sqrt{\frac{GM_\odot}{a^3}}. \tag{11.20}$$

Для гиперболической орбиты соответствующие уравнения имеют вид

$$\operatorname{sh} H_a = \frac{\sqrt{e^2 - 1}\sin\nu_a}{1 + e\cdot\cos\nu_a}, \tag{11.21}$$

$$M_a = e\cdot\operatorname{sh} H_a - H_a, \tag{11.22}$$

$$t_0 = t_a - M_a \Big/ \sqrt{GM_\odot / |a|^3}\,. \tag{11.23}$$

Метод вычисления элементов орбиты по двум ее точкам использован в функции Elements.

```
// Elements· Вычисление элементов орбиты по двум заданным положениям
//   GM - [AU^3*d^-2], произведение гравитационной постоянной на центральную массу
//   Mjd_a    Время прохождения положения А, МЮД
//   Mjd_b    Время прохождения положения В, МЮД
//   r_a      Гелиоцентрические эклиптические координаты A, a. e
//   r_b      Гелиоцентрические эклиптические координаты B, a  e.
//   Mjd_p    Время прохождения перигелия
//   q        Расстояние перигелия, a. e.
//   e        Эксцентриситет орбиты
//   i        Наклонение орбиты к эклиптике, рад
//   Omega    Долгота восходящего узла, рад
//   omega    Аргумент перигелия, рад
//------------------------------------------------------------------------------
void Elements ( double GM, double Mjd_a, double Mjd_b,
                const Vec3D& r_a, const Vec3D& r_b,
                double& Mjd_p, double& q, double& e,
                double& i, double& Omega, double& omega )
{

  double  tau, eta, p,
  double  a, n, nu, E, M, u;
  double  s_a, s_b, s_0, fac, sinhH,
  double  cos_dnu, sin_dnu, ecos_nu, esin_nu;
  Vec3D   e_a, r_0, e_0, R;

  // Вычисление вектора r_0 (составляющая r_b, перпендикулярная к r_a)
  // и величин r_a,r_b и r_0
  s_a = Norm(r_a);  e_a = r_a/s_a;
  s_b = Norm(r_b);
  fac = Dot(r_b,e_a); r_0 = r_b-fac*e_a;
  s_0 = Norm(r_0);  e_0 = r_0/s_0,

  // Наклонение и восходящий узел
  R     = Cross(e_a,e_0);
  i     = pi/2 0 - R[theta],
  Omega = Modulo ( pi/2 0 + R[phi], 2 0*pi ),

  if (i==0.0)
    u = atan2 ( r_a[y], r_a[x] );
  else
    u = atan2 ( (+e_0[x]*R[y]-e_0[y]*R[x]) , (-e_a[x]*R[y]+e_a[y]*R[x]) ),

  // Фокальный параметр
  tau = sqrt(GM) * fabs(Mjd_b-Mjd_a);
```

```
  eta = FindEta ( r_a, r_b, tau );
  p   = pow ( s_a*s_0*eta/tau, 2 );

  // Эксцентриситет, истинная аномалия и аргумент перигелия
  cos_dnu = fac / s_b;
  sin_dnu = s_0 / s_b;

  ecos_nu = p / s_a - 1.0;
  esin_nu = ( ecos_nu * cos_dnu - (p/s_b-1.0) ) / sin_dnu;

  e  = sqrt ( ecos_nu*ecos_nu + esin_nu*esin_nu );
  nu = atan2(esin_nu,ecos_nu);
  omega = Modulo(u-nu,2.0*pi);

  // Расстояние перигелия, большая полуось и среднее движение
  q = p/(1.0+e);
  a = q/(1.0-e);
  n = sqrt ( GM / fabs(a*a*a) );

  // Средняя аномалия и время прохождения перигелия
  if (e<1.0) {
    E = atan2 ( sqrt((1.0-e)*(1.0+e)) * esin_nu,  ecos_nu + e*e );
    M = E - e*sin(E);
  }
  else
  {
    sinhH = sqrt((e-1.0)*(e+1.0)) * esin_nu / ( e + e * ecos_nu );
    M = e * sinhH - log ( sinhH + sqrt(1.0+sinhH*sinhH) ),
  }

  Mjd_p = Mjd_a - M / n;
}
```

11.2. Упрощенный метод Гаусса

11.2.1. Геометрия наблюдений, отнесенных к центру Земли

Если вектор $\boldsymbol{r}$ представляет положение планеты относительно Солнца, а $\boldsymbol{R}$ — геоцентрические координаты Солнца, можно получить ее геоцентрические координаты

$$\rho \boldsymbol{e} = \boldsymbol{R} + \boldsymbol{r}. \tag{11.24}$$

Здесь ρ — расстояние планеты от Земли, а $\boldsymbol{e}$ — единичный вектор, направленный к ней от центра Земли. В эклиптической и экваториальной системах координат вектор $\boldsymbol{e}$ имеет следующие компоненты:

$$\begin{pmatrix} \cos\lambda\ \cos\beta \\ \sin\lambda\ \cos\beta \\ \sin\beta \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} \cos\alpha\ \cos\delta \\ \sin\alpha\ \cos\delta \\ \sin\delta \end{pmatrix},$$

где λ и β — эклиптические долгота и широта, а α и δ — прямое восхождение и склонение планеты, наблюдаемой с Земли.

Если определены координаты планеты (α и δ) на небесной сфере, значит, установлено наблюдаемое направление на нее $\boldsymbol{e}$. Расстояние ρ неизвестно и должно быть определено в процессе вычисления орбиты. Для однозначного решения задачи вычисления орбиты необходимы по крайней мере три наблюдения, из которых находятся направления $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$. Предположим, что на каждый момент наблюдения известны координаты Солнца $\boldsymbol{R}_1$, $\boldsymbol{R}_2$, $\boldsymbol{R}_3$. Теперь нужно вычислить расстояния ρ_1, ρ_2, ρ_3 на основе этих сведений. Только после этого можно будет определить положения точек, определяющих орбиту и ее элементы, относительно центра Солнца.

Геометрические соотношения, полученные из трех наблюдений, теперь должны быть выражены в форме, пригодной для вычисления орбиты.

В моменты времени $t_1 < t_2 < t_3$ планета находится в положениях $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3$ относительно Солнца. Поскольку при невозмущенном кеплеровском движении все точки и центр Солнца находятся в одной плоскости, всегда имеется возможность выразить один позиционный вектор через комбинацию двух других. Выберем $\boldsymbol{r}_2$ и запишем

$$\boldsymbol{r}_2 = n_1\boldsymbol{r}_1 + n_3\boldsymbol{r}_3 \quad \text{(уравнение плоскости орбиты).} \qquad (11.25)$$

Множители n_1 и n_3 зависят от взаимного расположения $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3$. Далее, если дуга орбиты, заключенная между крайними точками, составляет менее 180°, то оба множителя положительны. Комбинируя (11.24) и (11.25), получим

$$(\rho_2\boldsymbol{e}_2 - \boldsymbol{R}_2) = n_1 \cdot (\rho_1\boldsymbol{e}_1 - \boldsymbol{R}_1) + n_3 \cdot (\rho_3\boldsymbol{e}_3 - \boldsymbol{R}_3)$$

или после перестановки

$$n_1\rho_1\boldsymbol{e}_1 - \rho_2\boldsymbol{e}_2 + n_3\rho_3\boldsymbol{e}_3 = n_1\boldsymbol{R}_1 - \boldsymbol{R}_2 + n_3\boldsymbol{R}_3. \qquad (11.26)$$

Если мы зададим векторы

$$\boldsymbol{d}_1 = \boldsymbol{e}_2 \times \boldsymbol{e}_3 \quad \boldsymbol{d}_2 = \boldsymbol{e}_3 \times \boldsymbol{e}_1 \quad \boldsymbol{d}_3 = \boldsymbol{e}_1 \times \boldsymbol{e}_2,$$

то в силу свойств умножения вектор $\boldsymbol{d}_1$ перпендикулярен $\boldsymbol{e}_2$ и $\boldsymbol{e}_3$, вектор $\boldsymbol{d}_2$ перпендикулярен $\boldsymbol{e}_3$ и $\boldsymbol{e}_1$, а вектор $\boldsymbol{d}_3$ перпендикулярен $\boldsymbol{e}_1$ и $\boldsymbol{e}_2$. Следовательно, результат векторного произведения $\boldsymbol{e}_i \cdot \boldsymbol{d}_i$ отличен от нуля только при $i = j$. Векторное умножение выражения (11.26) на $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, $\boldsymbol{d}_3$ (по отдельности) дает следующие уравнения

$$n_1\rho_1 \cdot (\boldsymbol{e}_1 \cdot \boldsymbol{d}_1) = (n_1\boldsymbol{R}_1 - \boldsymbol{R}_2 + n_3\boldsymbol{R}_3) \cdot \boldsymbol{d}_1,$$
$$-\rho_2 \cdot (\boldsymbol{e}_2 \cdot \boldsymbol{d}_2) = (n_1\boldsymbol{R}_1 - \boldsymbol{R}_2 + n_3\boldsymbol{R}_3) \cdot \boldsymbol{d}_2,$$
$$n_3\rho_3 \cdot (\boldsymbol{e}_3 \cdot \boldsymbol{d}_3) = (n_1\boldsymbol{R}_1 - \boldsymbol{R}_2 + n_3\boldsymbol{R}_3) \cdot \boldsymbol{d}_3.$$

Вводя сокращения

$$D = \boldsymbol{e}_1 \cdot (\boldsymbol{e}_2 \times \boldsymbol{e}_3) = \boldsymbol{e}_2 \cdot (\boldsymbol{e}_3 \times \boldsymbol{e}_1) = \boldsymbol{e}_3 \cdot (\boldsymbol{e}_1 \times \boldsymbol{e}_2) = \boldsymbol{e}_1 \cdot \boldsymbol{d}_1 = \boldsymbol{e}_2 \cdot \boldsymbol{d}_2 = \boldsymbol{e}_3 \cdot \boldsymbol{d}_3$$

и

$$D_{ij} = \boldsymbol{d}_i \cdot \boldsymbol{R}_j,$$

получаем три уравнения

$$\rho_1 = \frac{1}{n_1 D}(n_1 D_{11} - D_{12} + n_3 D_{13}),$$
$$\rho_2 = \frac{1}{-D}(n_1 D_{21} - D_{22} + n_3 D_{23}), \qquad (11.27)$$
$$\rho_3 = \frac{1}{n_3 D}(n_1 D_{31} - D_{32} + n_3 D_{33}).$$

Следовательно, расстояния ρ_1, ρ_2 и ρ_3 могут быть выражены через n_1 и n_3 так же, как и векторы $\boldsymbol{e}_{1\ 3}$ и $\boldsymbol{R}_{1\ 3}$. На первый взгляд может показаться, что мы немногого этим добились, поскольку n_1 и n_3 нам неизвестны. Однако с помощью уравнения плоскости орбиты нам удалось уменьшить число неизвестных с трех ($\rho_{1\ 3}$) до двух ($n_{1,3}$). Вновь введенные коэффициенты полезны, в частности, тем, что, как сейчас будет показано, могут быть аппроксимированы выражениями, содержащими известные интервалы между наблюдениями.

Рассмотрим уравнение плоскости орбиты (11.25). Если вспомнить, что векторное произведение вектора на самого себя равно нулю, понятно, что векторное умножение обеих сторон уравнения плоскости орбиты на $\boldsymbol{r}_3$ или $\boldsymbol{r}_1$ дает следующие выражения

$$(\boldsymbol{r}_2 \times \boldsymbol{r}_3) = n_1 \cdot (\boldsymbol{r}_1 \times \boldsymbol{r}_3), \qquad (\boldsymbol{r}_1 \times \boldsymbol{r}_2) = n_3 \cdot (\boldsymbol{r}_1 \times \boldsymbol{r}_3)$$

и

$$n_1 = \frac{|\boldsymbol{r}_2 \times \boldsymbol{r}_3|}{|\boldsymbol{r}_1 \times \boldsymbol{r}_3|}, \qquad n_3 = \frac{|\boldsymbol{r}_1 \times \boldsymbol{r}_2|}{|\boldsymbol{r}_1 \times \boldsymbol{r}_3|}.$$

Так как площадь треугольника, ограниченного векторами $\boldsymbol{r}_a$ и $\boldsymbol{r}_b$,

$$\Delta = \frac{1}{2}|\boldsymbol{r}_a \times \boldsymbol{r}_b|,$$

n_1 и n_3 могут быть представлены в виде выражений для площадей треугольников, заключенных между $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3$ (см. рис. 11.3):

$$n_1 = \frac{\Delta_1}{\Delta_2}, \qquad n_3 = \frac{\Delta_3}{\Delta_2}.$$

Если дуги орбиты малы, то площади треугольников лишь незначительно отличаются от площадей соответствующих секторов $S_i = \eta_i \Delta_i$, которые пропорциональны интервалам τ_i:

$$n_1 = \frac{\eta_2}{\eta_1} \cdot \frac{\tau_1}{\tau_2} \approx \frac{\tau_1}{\tau_2}, \qquad n_3 = \frac{\eta_2}{\eta_3} \cdot \frac{\tau_3}{\tau_2} \approx \frac{\tau_3}{\tau_2}. \qquad (11.28)$$

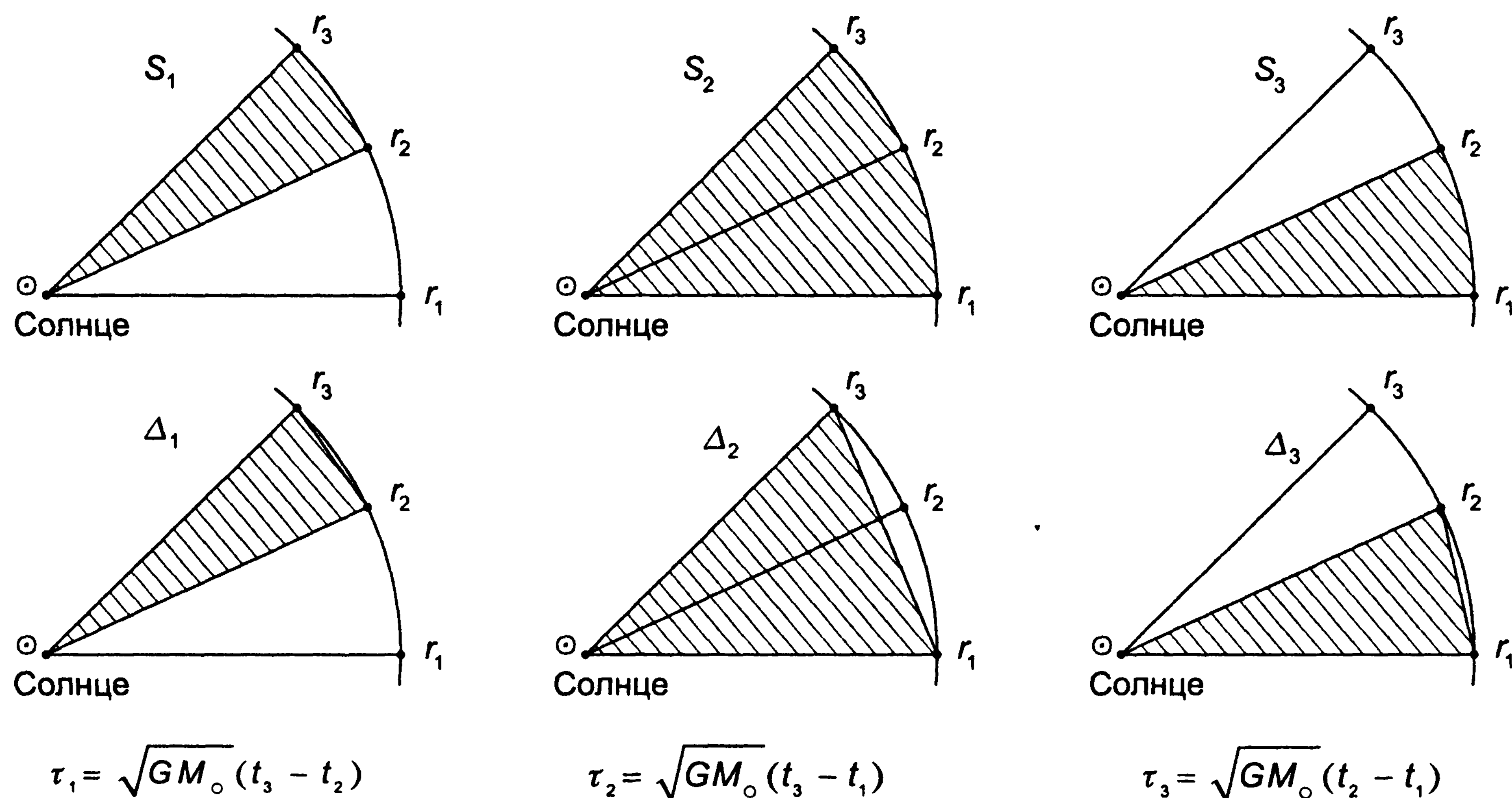


Рис. 11.3. Площади секторов, площади треугольников и интервалы для трех гелиоцентрических положений

Теперь мы знаем по крайней мере приближенные значения n_1 и n_3, позволяющие нам примерно определить геоцентрические расстояния (ρ_1, ρ_2, ρ_3).

11.2.2. Последовательное приближение для отношения площадей сектора и треугольника

В итоге подробного рассмотрения проблемы можно сделать два существенных вывода. Если известны два гелиоцентрических положения небесного тела в некоторые моменты времени, можно однозначно вычислить его орбиту. То же самое можно сказать и относительно элементов орбиты и отношения площадей сектора и треугольника. С другой стороны, если известна величина этого отношения для серии из трех наблюдений, можно вычислить позиционные векторы, описывающие геоцентрические и гелиоцентрические координаты. Упрощенный метод Гаусса для вычисления орбиты, который мы сейчас опишем, основан на этих выводах.

Пусть нам известны направления на светило из центра Земли (e_1, e_2, e_3), соответствующие геоцентрические координаты Солнца ($\boldsymbol{R}_1$, $\boldsymbol{R}_2$, $\boldsymbol{R}_3$) и интервалы (τ_1, τ_2, τ_3). Вычисления разделяются на следующие этапы:

1. Примем $n_1 = \tau_1/\tau_2$ и $n_3 = \tau_3/\tau_2$ за начальные приближения для отношений площадей треугольников.
2. Будем повторять шаги а–г до тех пор, пока значения n_1, n_3 не перестанут изменяться заметным образом.
   а. Вычислим геоцентрические расстояния (ρ_1, ρ_2, ρ_3) с помощью (11.27).
   б. Отсюда с помощью (11.24) получим геоцентрические позиционные векторы ($\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3$).

- в. Вычислим с помощью (11.5) соотношения площадей сектора и треугольника (η_1, η_2, η_3) для каждой пары позиционных векторов и соответствующих интервалов.
- г. Вычислим для отношений площадей сектора и треугольника уточненные значения величин $n_1 = (\eta_2/\eta_1)\cdot(\tau_1/\tau_2)$ и $n_3 = (\eta_2/\eta_3)\cdot(\tau_3/\tau_2)$.

3. Вычислим элементы орбиты по окончательным значениям r_1, r_3 и τ_2.

Вычисленная таким образом орбита обладает искомыми свойствами, а именно — тело, движущееся по ней в моменты времени t_1, t_2 и t_3 находится в точках $\boldsymbol{r}_1$, $\boldsymbol{r}_2$ и $\boldsymbol{r}_3$, которые соответствуют наблюдаемым направлениям $\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$. В принципе, задача вычисления орбиты решена. Однако на практике применение последовательного приближения для соотношений площадей сектора и треугольника имеет ряд существенных ограничений.

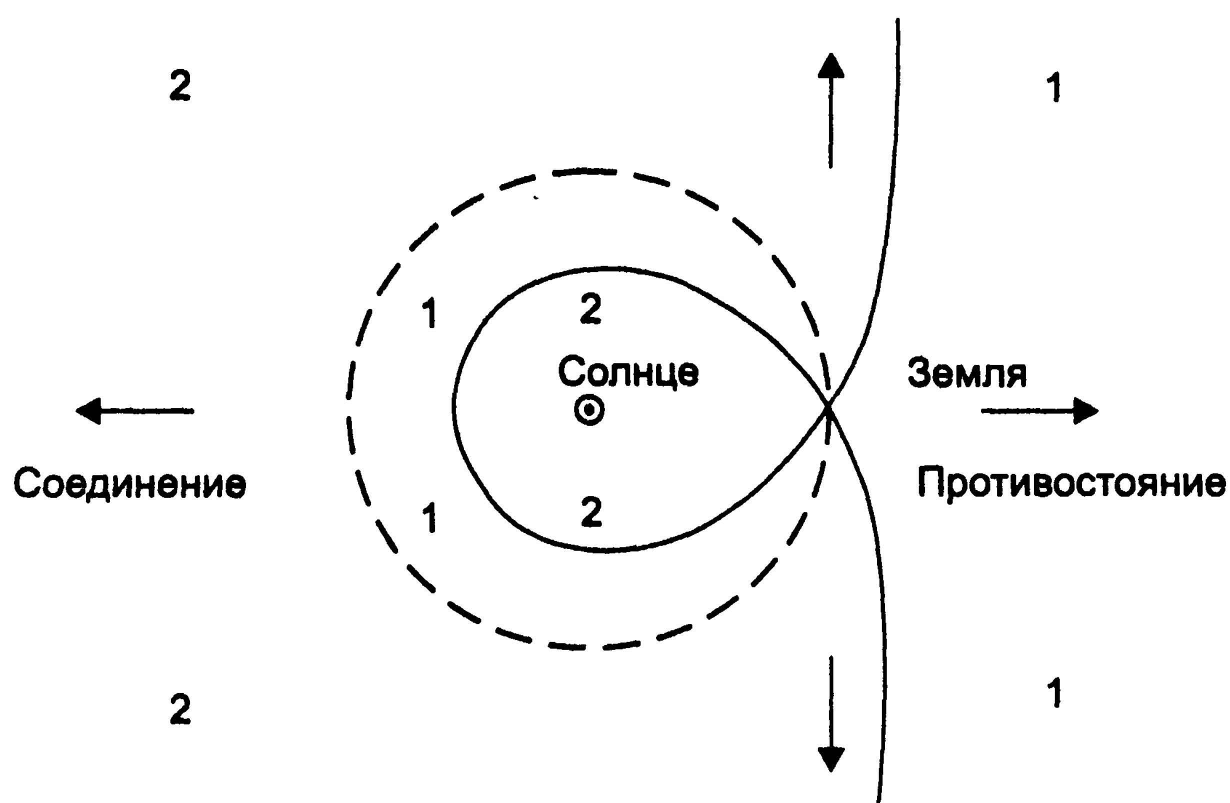


Рис. 11.4. Граничная линия Шарлье, разделяющая области, в которых орбита вычисляется однозначно (*1*) и неоднозначно (*2*)

11.2.3. Множественные решения

Упрощенный метод Гаусса всегда, если не считать исключительные случаи, когда итерация не сходится, обеспечивает результат, удовлетворяющий всем уравнениям, используемым при вычислении орбиты. В сомнительных случаях он может быть проверен последующим обратным вычислением по найденным элементам орбиты положений небесного тела, соответствующих трем наблюдениям. Однако не исключено, что иногда результат может показаться заведомо неверным или абсурдным даже на первый взгляд. Особенно это относится к случаям, явно противоречащим опыту, когда, например, вычисленная гиперболическая орбита обладает значительным эксцентриситетом[1]. Причина кроется в том, что вычисление орбит по трем наблюдениям оказывается не всегда единственным. В частности, это легко видеть на примере *решения, соответствующего орбите Земли*. Наблю-

[1] Эксцентриситет орбит всех без исключения известных комет менее $e = 1{,}1$.

датель движется по почти невозмущенной кеплеровской орбите вокруг Солнца и одновременно почти всегда находится в направлении наблюдения. Решение, соответствующее орбите Земли, легко распознается по значениям большой полуоси ($a \approx 1$), эксцентриситета ($e \approx 0$) и наклонения к эклиптике ($i \approx 0$).

Общие рассуждения о неоднозначности решения задачи вычисления орбиты обычно относятся к случаю малых дуг. Существуют четыре области, определяемые положением Солнца и орбитой Земли и разделенные так называемой граничной линией Шарлье (рис. 11.4), в которых решение либо единственно, либо нет. Например, в случае малой планеты, находящейся вблизи противостояния, решение единственно (за исключением упомянутого решения, соответствующего орбите Земли). С другой стороны, если комета наблюдается внутри каплевидной зоны вблизи Солнца или вблизи соединения за пределами земной орбиты, в принципе возможны два решения.

11.3. Полный метод Гаусса

Метод вычисления орбиты, изложенный в разделе 11.2.2, основан на методе последовательных приближений при вычислении отношений площадей треугольников n_1 и n_2 путем повторяющихся подстановок в соответствующие уравнения. Эта процедура не дает возможности судить о том, является ли решение единственным, или о способе нахождения других решений, если они существуют.

Общее представление о возможных решениях задачи вычисления орбиты может быть получено при некоторых упрощениях основных уравнений. Решения такой приближенной задачи, полученные с помощью уравнений Гаусса—Лагранжа, будут находиться в согласии с решениями строгих уравнений. Уравнения Гаусса—Лагранжа, будучи встроены в описанную выше процедуру вычисления орбит (см. раздел 11.2.2), позволят сделать обоснованный выбор и улучшить решения на каждом шаге итерации.

11.3.1. Уравнение Гаусса—Лагранжа

Прежде чем перейти к выводу уравнений Гаусса—Лагранжа, рассмотрим улучшенное приближение для отношений площадей треугольников, восходящее к Энке:

$$\begin{aligned} n_1 &= \frac{\tau_1}{\tau_2} + \frac{1}{6}\tau_1\tau_3\left(1 + \frac{\tau_1}{\tau_2}\right)\cdot\frac{1}{r_2^3}, \\ n_3 &= \frac{\tau_3}{\tau_2} + \frac{1}{6}\tau_1\tau_3\left(1 + \frac{\tau_3}{\tau_2}\right)\cdot\frac{1}{r_2^3}. \end{aligned} \qquad (11.29)$$

По сравнению с (11.28) приближение Энке дает лучшие результаты даже на больших интервалах, но зависит от остающегося неизвестным расстояния r_2.

Если теперь включить приближение Энке в соотношение (11.27) для геоцентрического расстояния, то, используя выражения

$$n_{10} = \frac{\tau_1}{\tau_2} \qquad \mu_1 = \frac{1}{6}\tau_1\tau_3\left(1 + \frac{\tau_1}{\tau_2}\right),$$

$$n_{30} = \frac{\tau_3}{\tau_2} \qquad \mu_3 = \frac{1}{6}\tau_1\tau_3\left(1 + \frac{\tau_3}{\tau_2}\right),$$

так же, как и

$$\rho_0 = -\frac{1}{D}(n_{10}D_{21} - D_{22} + n_{30}D_{23}) \qquad \sigma = +\frac{1}{D}(\mu_1 D_{21} + \mu_3 D_{23}),$$

мы получим простое соотношение

$$\rho_2 = \rho_0 - \frac{\sigma}{r_2^3} \tag{11.30}$$

между геоцентрическим расстоянием ρ_2 и гелиоцентрическим расстоянием $\boldsymbol{r}_2$ в момент второго наблюдения. Здесь ρ_0 и σ — известные величины, полученные непосредственно из наблюдений.

С другой стороны, мы получаем второе выражение для двух величин в уравнении $\boldsymbol{r}_2 = \rho_2\boldsymbol{e}_2 - \boldsymbol{R}_2$, ρ_2, $\boldsymbol{r}_2$ и $\boldsymbol{R}_2$ образуют стороны треугольника Солнце–Земля–планета. Извлекая корень, получаем из него соотношение

$$r_2 = \sqrt{(\rho_2 - \gamma R_2)^2 + R_2^2(1-\gamma^2)}, \tag{11.31}$$

в котором

$$\gamma = \boldsymbol{e}_2 \cdot \boldsymbol{R}_2 / R_2$$

обозначает косинус угла между наблюдаемым направлением $\boldsymbol{e}_2$ и направлением на Солнце $\boldsymbol{R}_2$. В противостоянии $\gamma = -1$, в соединении $\gamma = +1$.

Подстановкой в левую часть уравнения (11.31) значения $\boldsymbol{r}_2$, полученного из (11.30), получим уравнение Гаусса—Лагранжа

$$\sqrt[3]{\frac{\sigma}{\rho_0 - \rho_2}} = \sqrt{(\rho_2 - \gamma R_2)^2 + R_2^2(1-\gamma^2)}, \tag{11.32}$$

ключевое для определения неизвестных ρ_2 и r_2. Может быть до трех пар положительных величин, удовлетворяющих обоим уравнениям. Для каждого r_2 вычисляются отношения площадей треугольников n_1 и n_3, а также ρ_1 и ρ_3. Вместе с ρ_2 мы таким образом получаем гелиоцентрические координаты $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, и $\boldsymbol{r}_3$ и решаем задачу вычисления орбиты в рамках приближения Энке. Поскольку каждая пара (ρ_2, r_2) приводит к следующему значению величин $\boldsymbol{r}_{1,2,3}$, мы теперь можем получить до трех разных орбит, соответствующих наблюдениям.

Несмотря на существование системы уравнений (11.30, 11.31), в большинстве известных способов вычисления орбит используются другие, по существу, эквивалентные формы уравнения Гаусса—Лагранжа. Гаусс при вычислении орбиты

Цереры создал выражение в тригонометрической форме, подходившее для разработанной им процедуры. Уравнение восьмого порядка

$$r_2^8 - \left\{ \rho_0^2 - 2\gamma\rho_0 R_2 + R_2^2 \right\} r_2^6 + \left\{ 2\sigma(\rho_0 - \gamma R_2) \right\} r_2^2 - \left\{ \sigma^2 \right\} = 0 \qquad (11.33)$$

для гелиоцентрического расстояния r_2 восходит к Лагранжу. Оно получается при подстановке значения ρ_2, полученного из уравнения (11.30) в (11.31). При упрощении $\sigma \approx \rho_0 R_2^3$, допустимом лишь для очень малых отрезков орбиты, это уравнение восьмого порядка является ядром метода Лапласа для вычисления орбит.

Полезно для решения уравнения Гаусса—Лагранжа вначале посмотреть на график двух функций

$$\hat{r}(\rho) = \sqrt[3]{\frac{\sigma}{\rho_0 - \rho}} \qquad \text{и} \qquad \tilde{r}(\rho) = \sqrt{(\rho - \gamma R)^2 + R^2(1 - \gamma^2)},$$

пересечения которых дают искомые решения. Если функция $\hat{r}$ убывает или возрастает строго монотонно и имеет полюс при $\rho = \rho_0$, то график функции $\tilde{r}$ представляет собой обращенную вогнутостью вверх гиперболу, имеющую минимум при $\rho = \gamma R$. Из этого следует, что в принципе существуют одно или три пересечения двух графиков, положение которых (поскольку $R \approx 1$ а. е.) зависит только от σ, ρ_0 и γ.

При допущенном упрощении $\sigma \approx \rho_0 R_2^3$, которое допустимо лишь для малых дуг орбиты, всегда можно найти решение, соответствующее орбите Земли, когда $\rho = 0$ а. е. и $r = 1$ а. е., независимо от остальных величин. При этом упрощении существует самое большее два положительных решения для ρ, одно из которых представляет истинную орбиту наблюдаемого объекта. Интересно, что это упрощение не применимо в случае вычисления орбиты Цереры, рассмотренном Гауссом, поскольку здесь уравнение Гаусса—Лагранжа имеет единственное решение.

Для решения этого уравнения мы используем функцию SolveGL. Комбинирование двойной секущей и метода Ньютона позволяет найти все (до трех) решения (11.32), отбросить отрицательные, то есть не имеющие физического смысла, решения и отсортировать соответственно их значениям. Мы приводим здесь лишь спецификацию заголовка функции, из которой видны передаваемые параметры, поскольку различение случаев требует большого числа шагов, а также из-за большой длины кода.

```
//-----------------------------------------------------------------------------
// SolveGL  Finds the (up to three) Нахождение решений уравнения Гаусса-Лагранжа (до трех)
// Используются лишь положительные решения
//   gamma      Косинус элонгации (от Солнца) в момент второго наблюдения
//   R          Расстояние между Землей и Солнцем в момент второго наблюдения
//   rho_0      Нулевой порядок приближения для геоцентрического расстояния
//   sigma      Вспомогательная величина
//   rho_a      Геоцентрическое расстояние в момент второго наблюдения (первое решение)
//   rho_b      Геоцентрическое расстояние в момент второго наблюдения (второе решение)
//   rho_c      Геоцентрическое расстояние в момент второго наблюдения (третье решение)
//   n          Количество найденных решений
//-----------------------------------------------------------------------------
void SolveGL ( double gamma, double R, double rho_0, double sigma,
               double& rho_a, double& rho_b, double& rho_c, int& n )
```

11.3.2. Улучшенная итерация для отношения площадей треугольников

Приближение Энке не позволяет получить точное решение основной задачи вычисления орбит. Однако его можно так комбинировать с итерацией для отношения площадей треугольников в упрощенном методе Гаусса, что возможности методов существенно расширяются. В результате получится процесс для осуществления полного метода Гаусса, который мы сейчас опишем.

Вновь примем, что нам известны геоцентрические направления на светило ($\boldsymbol{e}_1$, $\boldsymbol{e}_2$, $\boldsymbol{e}_3$), соответствующие геоцентрические координаты Солнца ($\boldsymbol{R}_1$, $\boldsymbol{R}_2$, $\boldsymbol{R}_3$) и интервалы (τ_1, τ_2, τ_3). Используя эти исходные данные, проводим следующие этапы вычислений:

1. Примем $\mu_1 = (\tau_1\tau_3/6)(1 + \tau_1/\tau_2)$ и $\mu_3 = (\tau_1\tau_3/6)(1 + \tau_3/\tau_2)$ за начальные приближения для условий коррекции отношений площадей треугольников.
2. Будем повторять шаги а–д до тех пор, пока значения μ_1, μ_3 и остальных величин не перестанут изменяться заметным образом.
   а. Вычислим ρ_0 и σ и решим соответствующее уравнение Гаусса—Лагранжа. Выберем первое из трех возможных решений для ρ_2 и вычислим соответствующее гелиоцентрическое расстояние r_2.
   б. Вычислим отношения площадей сектора и треугольника $n_1 = n_{10} + \mu_1/r_2^3$ и $n_3 = n_{30} + \mu_3/r_2^3$ и с помощью (11.27) по найденным значениям получим геоцентрические расстояния (ρ_1, ρ_2, ρ_3).
   в. Вычислим гелиоцентрические позиционные векторы ($\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3$) с помощью (11.24).
   г. Вычислим с помощью (11.5) отношения площадей сектора и треугольника (η_1, η_2, η_3) для каждой пары позиционных векторов и соответствующих интервалов.
   д. Вычислим уточненные значения величин $\mu_1 = (\eta_2/\eta_1 - 1)\cdot(\tau_1/\tau_2)\cdot r_2^3$ и $\mu_3 = (\eta_2/\eta_3 - 1)\cdot(\tau_3/\tau_2)\cdot r_2^3$ для отношения площадей сектора и треугольника.
3. Вычислим элементы орбиты по окончательным значениям $\boldsymbol{r}_1$, $\boldsymbol{r}_3$ и τ_2.

Если уравнение Гаусса—Лагранжа имеет более одного решения, вся процедура повторяется. Если необходимо, полученное решение далее может быть исследовано на правдоподобие в случае решения, соответствующего орбите Земли или гиперболической орбите. В сомнительных случаях выбрать правильную орбиту среди нескольких решений поможет четвертое наблюдение.

11.3.3. Время распространения света

Для практического использования метода необходимо внести еще одно небольшое изменение. До сих пор мы предполагали, что наблюдаемое положение планеты или кометы всегда соответствует ее мгновенному геометрическому поло-

жению относительно Солнца и Земли. На самом деле существует некоторая погрешность, связанная с разницей во времени между испусканием света и его наблюдением, определяемая конечностью скорости света.

$c = 173{,}1$ а. е./сут
$1/c = 0^{d}{,}05776$/а. е.

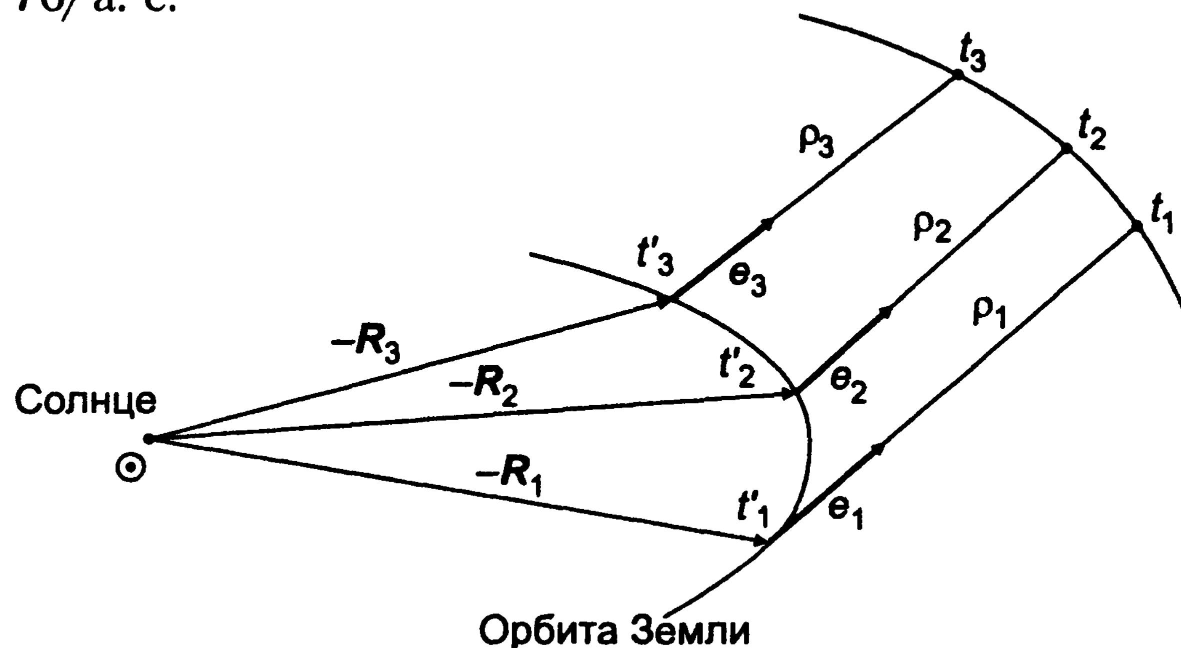


Рис. 11.5. Учет времени распространения света

Векторы e_i на рис. 11.5 указывают направления наблюдения и направлены из концов векторов $-R_i$, отмечающих положение Земли в момент наблюдения, к трем точкам $r_i = \rho_i e_i - R_i$, в которых планета находилась несколько ранее, в момент $t_i < t'_i$. Разность $t'_i - t_i$ соответствует времени, за которое луч света преодолевает расстояние ρ_i:

$$\rho_i = c \cdot (t'_i - t_i).$$

Возникает новая проблема, поскольку моменты времени испускания света t_i, знание которых требуется для вычисления интервалов τ_i, нам неизвестны. При вычислении орбиты, впрочем, мы получаем значения геоцентрических расстояний (ρ_1, ρ_2, ρ_3) со все возрастающей точностью, поэтому можно считать, что эффект, связанный с конечностью скорости света, не создает дополнительных трудностей. Время наблюдения t'_i используется лишь при определении начальных приближений для отношений площадей треугольников. Впоследствии, когда становятся известны действительные геоцентрические расстояния, интервалы

$$\tau_1 = \sqrt{GM_{\odot}}(t_3 - t_2), \qquad \tau_2 = \sqrt{GM_{\odot}}(t_3 - t_1), \qquad \tau_3 = \sqrt{GM_{\odot}}(t_2 - t_1)$$

вновь вычисляются при каждой итерации, причем

$$t_i = t'_i - \rho_i \cdot 0^{d}{,}05776/\text{а. е.} \qquad (i = 1 \ldots 3),$$

Вспомогательные геометрические переменные d_i, D_{ij}, и D зависят только от направления векторов e_i и координат Солнца R_i, поэтому не требуют поправок.

11.4. Программа Gauss

Приведенная далее программа Gauss позволяет вычислить орбиту кометы или астероида на основании трех наблюдений. Вначале по прямому восхождению

и склонению определяются единичные векторы направления на светило и координаты Солнца (Start). Поскольку искомые элементы орбиты обычно относят к эклиптике, все величины вычисляются в эклиптической системе координат при произвольно выбранной эпохе. Собственно вычисление орбиты производится функцией GaussMethod с помощью полного метода Гаусса. Для решения уравнения Гаусса—Лагранжа используется функция SolveGL, которая ради экономии места приводится в сокращенном виде.

Большое количество данных вынуждает к тому, чтобы они были собраны в файл ввода **Gauss.dat**. Файл данных начинается с единственной строки комментария, позволяющей распознать эти данные. Три последующие строки содержат дату (год, месяц, число и час в виде десятичной дроби), наблюденные экваториальные координаты (прямое восхождение —часы, минуты и секунды в виде десятичной дроби; склонение — градусы, минуты и секунды в виде десятичной дроби), а также комментарий соответствующего действия. Далее в двух строках указываются эпоха, к которой относятся наблюдаемые координаты, и эпоха, к которой должны быть отнесены искомые элементы орбиты. Следующий пример содержит данные наблюдений Цереры, использованные Гауссом для вычисления ее орбиты.

```
Ceres (Gauss's example)
 1805 09 05 24 165   6 23 57.54  22 21 27 08  ! Three observations
 1806 01 17 22.095   6 45 14 69  30 21 24 20  ! Format· Date (ymdh).
 1806 05 23 20 399   8 07 44 60  28 02 47.04  !        RA (hms). Dec (hms)
 1806.0                                       ! Equinox
 1806.0                                       ! Required equinox
```

При вычислении орбит все данные наблюдений должны быть представлены в виде астрометрических координат, то есть освобождены от влияния аберрации света. Это имеет значение во всех случаях, когда координаты светила определяются по отношению к окружающим звездам, выбранным из каталога. Примером тому является измерение фотографий звездных полей — наиболее распространенный метод определения пути кометы или астероида.

```
//------------------------------------------------------------------------
// Модуль·     Программа Gauss (Gauss.cpp)
// Задача:   Вычисление орбиты по трем наблюдениям
// (c) 1999 Oliver Montenbruck. Thomas Pfleger
//------------------------------------------------------------------------

#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Const.h"
#include "APC_IO.h"
#include "APC_Kepler.h"
#include "APC_Math.h"
#include "APC_PrecNut.h"
#include "APC_Spheric.h"
#include "APC_Sun.h"
#include "APC_Time h"
```

```
#include "APC_VecMat3D.h"

#ifdef __GNUC__  // адаптация для GNU C++
#include "GNU_iomanip.h"
#endif

using namespace std;

//------------------------------------------------------------------------------
// Start: Чтение файла ввода и подготовка данных наблюдений
//   InputFile   Имя файла ввода
//   Header      Комментарий
//   R_Sun[]     Три вектора эклиптических координат Солнца
//   e[]         Три единичных вектора, указывающих направление на наблюдаемый объект
//   Mjd0[]      Три момента времени наблюдения, модифицированная юлианская дата
//   T_eqx       Эпоха R_Sun и e в юлианских столетиях, прошедших после J2000
//------------------------------------------------------------------------------
void Start ( char* InputFile,
             char* Header,
             Vec3D R_Sun[], Vec3D e[], double Mjd0[], double& T_eqx )
{
  //
  // Variables
  //
  int       i, Year, Month, Day, deg, min;
  double    Hour, secs, year;
  double    T_eqx0, T, RA, Dec;
  ifstream  inp;

  inp.open(InputFile);  // Открытие файла данных наблюдений
  inp.get(Header,81);   // Чтение заголовка

  // Чтение данных наблюдений
  for (i=0;i<=2;i++) {

    inp >> Year >> Month >> Day >> Hour;
    inp >> deg >> min >> secs;
    RA = Rad*15.0*Ddd(deg,min,secs);
    inp >> deg >> min >> secs;
    Dec = Rad*Ddd(deg,min,secs);
    inp.ignore(81,'\n');
    Mjd0[i] = Mjd(Year,Month,Day) + Hour/24.0;
    e[i]    = Vec3D(Polar(RA,Dec));
  }

  // Эпоха наблюдения и желаемая эпоха для элементов орбиты
  inp >> year; inp.ignore(81,'\n');
  T_eqx0 = (year-2000.0)/100.0;
  inp >> year; inp.ignore(81,'\n');
  T_eqx = (year-2000.0)/100.0;

  // Начальные данные для вычисления орбиты (данные наблюдений
  // и координаты Солнца, отнесенные к эклиптике и равноденствию даты)
  for (i=0;i<=2;i++) {
    T        = (Mjd0[i]-MJD_J2000) / 36525.0;
    e[i]     = PrecMatrix_Ecl(T_eqx0,T_eqx) * Equ2EclMatrix(T_eqx0) * e[i];
    R_Sun[i] = PrecMatrix_Ecl(T,T_eqx) * SunPos(T);
```

```
  }

  // Печать исходных данных
  cout << " Summary of orbit determination" << endl << endl
       << "  " << Header << endl  << endl
       << " Initial data (geocentric ecliptic coordinates; Equinox J"
       << fixed << setprecision(2) << 100.0*T_eqx+2000.0 << ")" << endl
       << endl;
  cout << "  Observation"
       << setw(25) << "1" << setw(12) << "2" << setw(12) << "3" << endl;
  cout << "  Julian Date                ";
  for (i=0;i<=2;i++) { cout << setw(12) << Mjd0[i]+2400000.5; };
  cout << endl;
  cout << "  Solar longitude [deg]       " << setprecision(4);
  for (i=0;i<=2;i++) { cout << setw(12) << Deg*(R_Sun[i][phi]); };
  cout << endl;
  cout << "  Planet/Comet longitude [deg]";
  for (i=0;i<=2;i++) { cout << setw(12) << Deg*((e[i])[phi]); };
  cout << endl;
  cout << "  Planet/Comet latitude [deg] ";
  for (i=0;i<=2;i++) { cout << setw(12) << Deg*((e[i])[theta]); };
  cout << endl;

}


//------------------------------------------------------------------------------
// DumpElem: Форматный вывод элементов орбиты
//   Mjd_p   Модифицированная юлианская дата прохождения перигелия
//   q       Расстояние перигелия, а.е.
//   e       Эксцентриситет
//   i       Наклонение, рад
//   Omega   Долгота восходящего узла, рад
//   omega   Аргумент перигелия, рад
//   T_eqx   Соответствующая эпоха в юлианских столетиях, прошедших после J2000
//------------------------------------------------------------------------------
void DumpElem ( double Mjd_p, double q, double e, double i,
                double Omega, double omega, double T_eqx )
{ ... }

//------------------------------------------------------------------------------
// Retard: Поправка за время распространения света и вычисление разницы моментов времени
//   Mjd0[]  Моменты наблюдения (t1',t2',t3') (модифицированная юлианская дата)
//   rho[]   Три геоцентрических расстояния, а. е.
//   Mjd[]   Моменты испускания света (t1,t2,t3) (модифицированная юлианская дата)
//   tau[]   Масштабированные разности моментов времени
//------------------------------------------------------------------------------
void Retard ( const double Mjd0[], const double rho[],
              double Mjd[], double tau[] )
{
  for (int i=0;i<=2;i++)  Mjd[i] = Mjd0[i] - rho[i]/c_light;

  tau[0] = kGauss * (Mjd[2]-Mjd[1]);
  tau[1] = kGauss * (Mjd[2]-Mjd[0]);
  tau[2] = kGauss * (Mjd[1]-Mjd[0]);
}
```

```
//------------------------------------------------------------------------------
// SolveGL: Нахождение решений (до трех) уравнения Гаусса-Лагранжа.
// Возвращаются только положительные решения.
//   gamma  Косинус элонгации в момент второго наблюдения
//   R      Расстояние от Земли до Солнца в момент второго наблюдения
//   rho_0  Нулевой порядок приближения для геоцентрического расстояния
//   sigma  Вспомогательная величина
//   rho_a  Геоцентрическое расстояние в момент второго наблюдения (1-е решение)
//   rho_b  Геоцентрическое расстояние в момент второго наблюдения (2-е решение)
//   rho_c  Геоцентрическое расстояние в момент второго наблюдения (3-е решение)
//   n      Количество действительных найденных решений
//------------------------------------------------------------------------------
void SolveGL ( double gamma, double R, double rho_0, double sigma,
               double& rho_a, double& rho_b, double& rho_c, int& n )
{
 ...
}


//------------------------------------------------------------------------------
//
// Select: Выбор единственного решения уравнения Гаусса-Лагранжа для ро
//   rho_a  Первое решение
//   rho_b  Второе решение
//   rho_c  Третье решение
//   n_sol  Общее количество решений
//   n      Номер выбранного решения
// Возвращаемые значения:  Выбранное решение для ро
//------------------------------------------------------------------------------
double Select ( double rho_a, double rho_b, double rho_c,
                int n_sol, int n )
{
  if ( (n<1) || (n_sol<n) ) {
    cerr << " Error in Select: n = " << setw(2) << n << " n_sol = "
         << setw(2) << n_sol << endl;
    exit(1);
  }
  else
    switch(n) {
      case 1: return rho_a;
      case 2: return rho_b;
      case 3: return rho_c;
    };
  return 0.0;
}


//------------------------------------------------------------------------------
// GaussMethod: Вычисление орбиты методом Гаусса
//   R_Sun[]   Три вектора эклиптических координат Солнца
//   e[]       Три единичных вектора, указывающих направление на объект
//   Mjd0[]    Три момента времени наблюдений (модифицированная юлианская дата)
//   N_run     Номер текущего запуска (выбранный номер решения уравнения Гаусса-Лагранжа)
//   N_sol     Общее количество решений уравнения Гаусса-Лагранжа
//   Mjd_p     Модифицированная юлианская дата прохождения перигелия
//   q         Расстояние перигелия, а.е.
//   ecc       Эксцентриситет
```

```
//   inc        Наклонение, рад
//   Omega      Долгота восходящего узла, рад
//   omega      Аргумент перигелия, рад
//----------------------------------------------------------------------------
void GaussMethod ( const Vec3D R_Sun[], const Vec3D e[], const double Mjd0[],
                   int& N_run, int& N_sol,
                   double& Mjd_p, double& q, double& ecc,
                   double& inc, double& Omega, double& omega )
{
  const double eps_rho = 1.0e-8;
  const int    maxit   = 20;
  int     i,j,iterat;
  double  rho_old;
  double  Det,l,L,gamma;
  double  rho_max,rho_mean,rho_min;
  double  Mjd[3],rho[3],n[3],n0[3],tau[3],eta[3];
  double  mu[3];
  Vec3D   d[3],r[3];
  double  D[3][3];
  // Матрица D и определитель Det
  d[0] = Cross(e[1],e[2]);
  d[1] = Cross(e[2],e[0]);
  d[2] = Cross(e[0],e[1]);
  for (i=0;i<=2;i++) for (j=0;j<=2;j++)  D[i][j] = Dot(d[i],R_Sun[j]);
  Det = Dot(e[2],d[2]);
  // Косинус угла наблюдаемого направления относительно Солнца
  // направление в момент второго наблюдения
  gamma  = Dot(e[1],R_Sun[1])/Norm(R_Sun[1]);
  // Разница во времени tau[i] и начальное приближение для мю
  tau[0] = kGauss*(Mjd0[2]-Mjd0[1]);
  tau[1] = kGauss*(Mjd0[2]-Mjd0[0]);
  tau[2] = kGauss*(Mjd0[1]-Mjd0[0]);
  mu[0] = (1.0/6.0) * tau[0]*tau[2] * (1.0+tau[0]/tau[1]);
  mu[2] = (1.0/6.0) * tau[0]*tau[2] * (1.0+tau[2]/tau[1]);
  // Header
  cout << endl << endl
       << " Solution No." << setw(1) << N_run << endl << endl
       << " Iteration of the geocentric distances " << endl
       << " Solutions (rho_2 in [AU]) of the Gauss-Lagrangian equation"
       << endl << endl;
  cout << "  Iteration             Number   1st Sol.    2nd Sol.    3rd Sol."
       << endl;

  // Итерационное уточнение tau[i] и mu[i]
  rho[1] = 0.0;
  iterat = 0;

  do {

    // Сохранение предыдущего значения и приращения итерации
    rho_old = rho[1];
    iterat++;

    // Определение геоцентрического расстояния ро в момент второго наблюдения
    // из уравнения Гаусса-Лагранжа
    n0[0] = tau[0]/tau[1];
    n0[2] = tau[2]/tau[1];
```

```
    L = - ( n0[0]*D[1][0]-D[1][1]+n0[2]*D[1][2] ) / Det;
    l =   ( mu[0]*D[1][0]    +    mu[2]*D[1][2] ) / Det;

    SolveGL (gamma,Norm(R_Sun[1]),L,l,rho_min,rho_mean,rho_max,N_sol);

    rho[1] = Select ( rho_min,rho_mean,rho_max, N_sol,N_run );
    r[1]   = rho[1]*e[1] - R_Sun[1];

    cout << setw(4) << iterat << setw(23) << N_sol << fixed << setprecision(8)
         << setw(15) << rho_min << setw(12) << rho_mean
         << setw(12) << rho_max << endl;

    // Вычисление n1 и n3
    n[0] = n0[0] + mu[0]/pow(Norm(r[1]),3);
    n[2] = n0[2] + mu[2]/pow(Norm(r[1]),3);

    // Геоцентрические расстояния rho_1 и rho_3 от n_1 и n_3
    rho[0] = ( n[0]*D[0][0] - D[0][1] + n[2]*D[0][2] ) / (n[0]*Det);
    rho[2] = ( n[0]*D[2][0] - D[2][1] + n[2]*D[2][2] ) / (n[2]*Det);

    // Поправка за время распространения света и вычисление разницы во времени
    Retard ( Mjd0, rho, Mjd, tau );

    // Вектор гелиоцентрических координат
    for (i=0;i<=2;i++)
      r[i] = rho[i]*e[i] - R_Sun[i];

    // Отношение площадей сектора и треугольника eta_i
    eta[0] = FindEta ( r[1],r[2], tau[0] );
    eta[1] = FindEta ( r[0],r[2], tau[1] );
    eta[2] = FindEta ( r[0],r[1], tau[2] );

    // Уточненные значения mu_1, mu_3
    mu[0] = ( eta[1]/eta[0] - 1.0 ) * (tau[0]/tau[1]) * pow(Norm(r[1]),3);
    mu[2] = ( eta[1]/eta[2] - 1.0 ) * (tau[2]/tau[1]) * pow(Norm(r[1]),3);

    if (maxit<=iterat) {
      cerr << "   (Convergence problems; iteration stopped after "
           << setw(2) << maxit << " steps)"
           << endl;
      break;
    }
  }
  while (fabs(rho[1]-rho_old) > eps_rho);

  cout << endl
       << " Geocentric and heliocentric distances" << endl
       << endl;
  cout << "  Observation"
       << setw(25) << "1" << setw(12) << "2" << setw(12) << "3" << endl;
  cout << "  rho [AU]" << setw(20) << " " << fixed << setprecision(8);
  for (i=0;i<=2;i++) cout << setw(12) << rho[i]; cout << endl;
  cout << "  r [AU]" << setw(22) << " " << fixed << setprecision(8);
  for (i=0;i<=2;i++) cout << setw(12) << Norm(r[i]); cout << endl;

  // Получение элементов орбиты из первого и третьего наблюдений
  Elements ( GM_Sun, Mjd[0], Mjd[2], r[0], r[2],
```

```
            Mjd_p, q, ecc, inc, Omega, omega );
}


//------------------------------------------------------------------------------
//
// Основная программа
//
//------------------------------------------------------------------------------

void main(int argc, char* argv[])
{
  //
  // Переменные
  //
  char      Header[81];
  int       N_run, N_sol;
  double    T_eqx;
  double    Mjd0[3];
  Vec3D     R_Sun[3],e[3];
  double    Mjd_p, q, ecc, inc, Omega, omega;
  char      InputFile[APC_MaxFilename] = "";
  char      OutputFile[APC_MaxFilename] = "";
  bool      FoundInputfile = false;
  bool      FoundOutputfile = false;
  ofstream  OutFile;

  // Название
  cout << endl
       << "      GAUSS: orbit determination from three observations "
       << endl
       << "             (c) 1999 Oliver Montenbruck, Thomas Pfleger "
       << endl << endl;

  // Поиск входных и необязательных выходных файлов
  GetFilenames( argc, argv, "Gauss.dat", InputFile, FoundInputfile,
                OutputFile, FoundOutputfile );

  // Прерывание программы, если файл ввода не может быть найден
  if (!FoundInputfile) {
    cerr << " Terminating program." << endl;
    exit(-1);
  }

  // Перенаправление вывода в файл
  if (FoundOutputfile) {
    OutFile.open(OutputFile);
    if (OutFile.is_open())
      cout = OutFile;
  }

  // Инициализация
  Start ( InputFile, Header, R_Sun, e, Mjd0, T_eqx );

  // Обработка всех решений
  N_run = 0;
  N_sol = 1;
```

```
  do {

    N_run++;

    // Вычисление орбиты методом Гаусса для решения N_run уравнения Гаусса-Лагранжа
    GaussMethod ( R_Sun, e, Mjd0, N_run, N_sol, Mjd_p,q,ecc,inc,Omega,omega );

    // Печать элементов орбиты
    DumpElem ( Mjd_p,q,ecc,inc,Omega,omega, T_eqx );

  }
  while (N_run<N_sol);

  if (OutFile.is_open()) OutFile.close();
}
```

После обращения к программе Gauss пользователю не нужно вводить дополнительные данные, поскольку все необходимые сведения содержатся в файле **Gauss.dat**. Для приведенных выше параметров получаем следующую распечатку:

```
     GAUSS: orbit determination from three observations
            (c) 1999 Oliver Montenbruck, Thomas Pfleger

 Using default input file Gauss.dat
 Summary of orbit determination

  Ceres (Gauss's example)

 Initial data (geocentric ecliptic coordinates; Equinox J1806.00)

  Observation                        1            2            3
  Julian Date                   2380570.51   2380704.42   2380830.35
  Solar longitude [deg]           162.9166     297.2126      61.9405
  Planet/Comet longitude [deg]     95.5389      99.8239     118.1018
  Planet/Comet latitude [deg]      -0.9918       7.2775       7.6473


 Solution No.1

 Iteration of the geocentric distances
 Solutions (rho_2 in [AU]) of the Gauss-Lagrangian equation
  Iteration             Number   1st Sol.     2nd Sol.     3rd Sol.
   1                      1     1.63338760   0.00000000   0.00000000
   2                      1     1.63476170   0.00000000   0.00000000
   3                      1     1.63599887   0.00000000   0.00000000
   4                      1     1.63655182   0.00000000   0.00000000
   5                      1     1.63676833   0.00000000   0.00000000
   6                      1     1.63684968   0.00000000   0.00000000
   7                      1     1.63687981   0.00000000   0.00000000
   8                      1     1.63689091   0.00000000   0.00000000
   9                      1     1.63689499   0.00000000   0.00000000
  10                      1     1.63689649   0.00000000   0.00000000
  11                      1     1.63689704   0.00000000   0.00000000
  12                      1     1.63689724   0.00000000   0.00000000
  13                      1     1.63689731   0.00000000   0.00000000
  14                      1     1.63689734   0.00000000   0.00000000
  15                      1     1.63689735   0.00000000   0.00000000
```

```
Geocentric and heliocentric distances

 Observation                      1          2          3
 rho [AU]                   2.90181811 1.63689735 2.95876464
 r [AU]                     2.68083076 2.58786985 2.54398416

Orbital elements (Equinox J1806.00)

 Perihelion date     tp      1806/06/28.039  (JD 2380865.539)
 Perihelion distance q [AU]    2.541676
 Semi-major axis     a [AU]    2.767165
 Eccentricity        e         0.081487
 Inclination         i        10.6178 deg
 Ascending node      Omega    80.9788 deg
 Long. of perihelion pi      147.0173 deg
 Arg. of perihelion  omega    66.0385 deg
```

Как можно заметить по значению эклиптических координат Солнца и Цереры, астероид в период наблюдений находился вблизи противостояния. В этом случае отпадает возможность двоякого решения задачи вычисления орбиты.

В качестве примера вычисления орбиты, когда возможны два нетривиальных положительных решения уравнения Гаусса—Лагранжа, рассмотрим три наблюдения кометы Оркиша (Orkisz) (C/1925 G1) в апреле 1925:

```
Comet Orkisz (Example from Stracke: Bahnbestimmung, p.182)
 1925 04 05  2.786  22 26 43.51  16 37 16.00  ! Three observations
 1925 04 08  2.731  22 29 42.90  19 46 25.10  ! Format: Date (ymdh),
 1925 04 11  2.614  22 32 55.00  23 04 52.30  !         RA (hms), Dec (dms)
 1925.0                                       ! Equinox
 1925.0                                       ! Required equinox
```

Запустив программу Gauss с файлом данных **Orkisz.dat** (см. раздел П.1.3), получим следующий результат:

```
     GAUSS: orbit determination from three observations
            (c) 1999 Oliver Montenbruck, Thomas Pfleger

Summary of orbit determination

 Comet Orkisz (Example from Stracke: Bahnbestimmung, p.182)

Initial data (geocentric ecliptic coordinates: Equinox J1925.00)

 Observation                      1           2           3
 Julian Date                 2424245.62  2424248.61  2424251.61
 Solar longitude [deg]          14.8175     17.7642     20.7031
 Planet/Comet longitude [deg]  345.0988    347.2423    349.5855
 Planet/Comet latitude [deg]    24.4159     27.0070     29.6985


Solution No.1

Iteration of the geocentric distances
Solutions (rho_2 in [AU]) of the Gauss-Lagrangian equation

 Iteration            Number  1st Sol.    2nd Sol.    3rd Sol.
```

```
 1                     2    1.67288439  6.37592623 -0.00141974
 2                     2    1.67314456  6.37552347 -0.00162268
 3                     2    1.67314511  6.37552331 -0.00162312
 4                     2    1.67314510  6.37552331 -0.00162312

Geocentric and heliocentric distances

 Observation                     1           2           3
 rho [AU]                   1.71490811  1.67314510  1.63322160
 r [AU]                     1.10821587  1.10918577  1.11233811

Orbital elements (Equinox J1925.00)

 Perihelion date     tp      1925/04/05.280  (JD 2424245.780)
 Perihelion distance q [AU]    1.108212
 Semi-major axis     a [AU]  -80.901104
 Eccentricity        e         1.013698
 Inclination         i       101.2244 deg
 Ascending node      Omega   318.9892 deg
 Long. of perihelion pi      359.8990 deg
 Arg. of perihelion  omega    40.9098 deg


Solution No.2

Iteration of the geocentric distances
Solutions (rho_2 in [AU]) of the Gauss-Lagrangian equation

 Iteration            Number   1st Sol.    2nd Sol.    3rd Sol.
 1                     2    1.67288439  6.37592623 -0.00141974
 2                     2    1.67368234  6.37178371 -0.00197746
 3                     2    1.67368197  6.37178566 -0.00197720
 4                     2    1.67368197  6.37178566 -0.00197720

Geocentric and heliocentric distances (Геоцентрическое и гелиоцентрическое расстояние)

 Observation                     1           2           3
 rho [AU]                   6.53717259  6.37178566  6.22572525
 r [AU]                     5.77840936  5.63979853  5.52113670

Orbital elements (Equinox J1925.00)

 Perihelion date      tp      1925/04/25.025  (JD 2424265.525)
 Perihelion distance q [AU]    5.262900
 Semi-major axis     a [AU]   -0.020764
 Eccentricity        e       254.460587
 Inclination         i        67.3802 deg
 Ascending node      Omega   326.7908 deg
 Long. of perihelion pi       21.6630 deg
 Arg. of perihelion  omega    54.8721 deg
```

Здесь уравнение Гаусса—Лагранжа имеет три решения, из которых одно ($\rho_2 \approx 0{,}0$ а. е.) представляет собой решение, соответствующее орбите Земли. Остающиеся решения ($\rho_2 \approx 1{,}7$ а. е.) и ($\rho_2 \approx 6{,}4$ а. е.) соответствуют физически возможным кометным орбитам. Неправдоподобное второе решение ($e \gg 1$) может быть сразу отвергнуто.

12. Астрометрия

Фотография дает сравнительно простой способ определения положения кометы или планеты относительно звезд с известными координатами. Все, что для этого требуется помимо самой фотопластинки, — это звездный каталог и, по возможности, атлас, с помощью которого легко произвести отождествление близких к объекту звезд.

На рис. 12.1 схематически представлена реальная фотография кометы Бредфилда (C/1972 E1), полученная в 1982 году. Сравнение с картой атласа показывает, что изображение покрывает область неба размером $\pm 4^m$ по прямому восхождению и $\pm 1°$ по склонению, а его центр имеет координаты ($\alpha_{2000} = 12^h 28^m 45^s$, $\delta_{2000} - 44°00'$). Кроме самой кометы на снимке видны звезды, координаты которых можно найти в каталоге PPM. Каталог охватывает северное и южное небо и содержит координаты около 470 000 звезд. Среднее расстояние между соседними звездами составляет около половины градуса. Для определения положения объекта на фотографии требуются по меньшей мере три опорных звезды с известными координатами. Чтобы получить наиболее точные данные, следует попытаться определить искомые координаты по отношению к наибольшему возможному числу окружающих звезд. Вначале мы рассмотрим, как участок неба отображается на фотопластинке.

12.1. Фотографическое изображение

Идеальная камера или объектив телескопа собирают лучи света от звезды в точку, находящуюся за объективом на расстоянии, известном под названием фокусного расстояния. Точка P, в которую отображается звезда, определяется проекцией луча, идущего от звезды, через центр объектива.

В системе координат, заданной $\boldsymbol{u}$, $\boldsymbol{v}$ и $\boldsymbol{w}$, вектор

$$\boldsymbol{e} = \begin{pmatrix} \cos(\delta)\cos(\alpha - \alpha_0) \\ \cos(\delta)\sin(\alpha - \alpha_0) \\ \sin(\delta) \end{pmatrix}$$

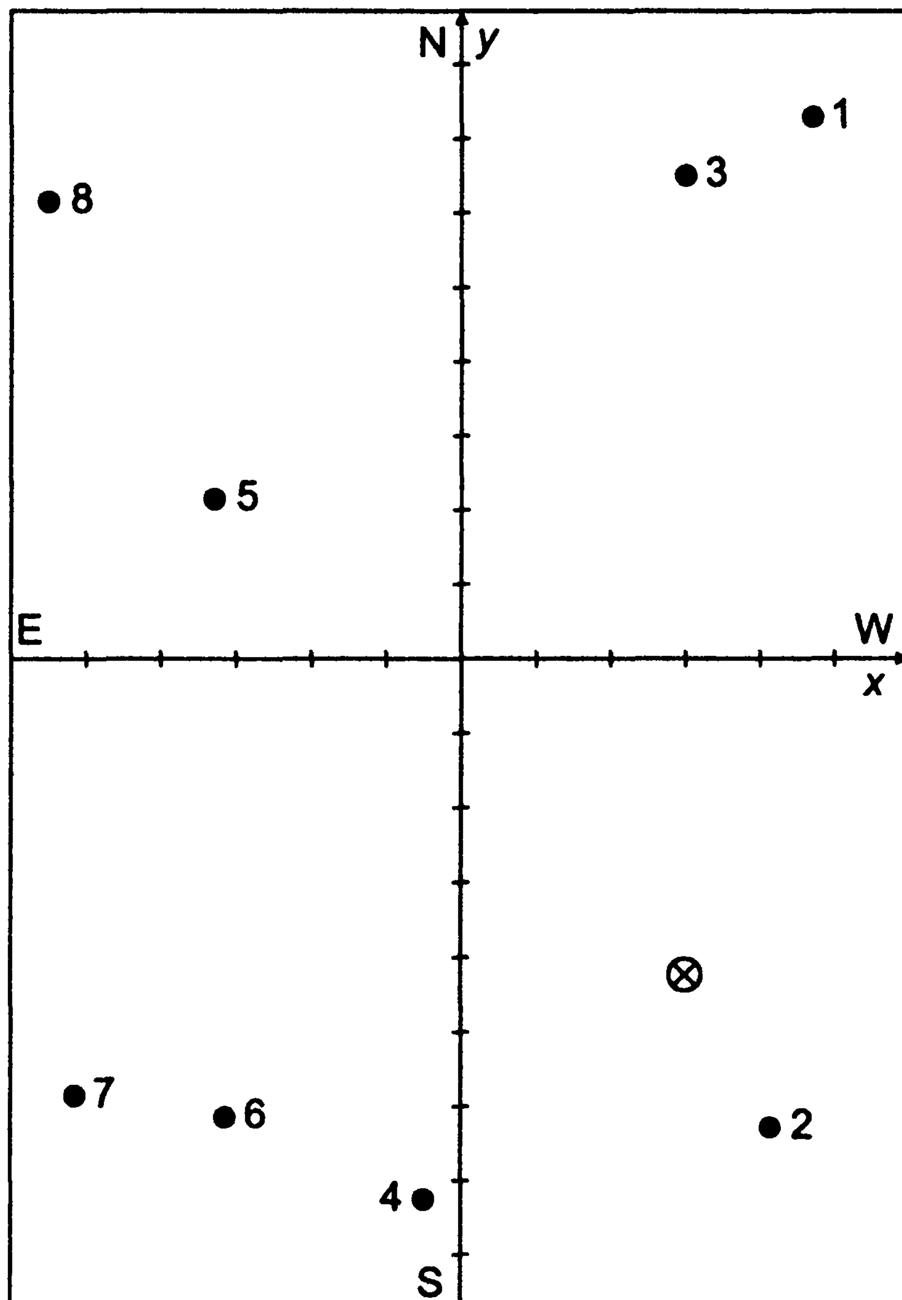
указывает направление на звезду с координатами α по прямому восхождению и δ по склонению.

Соответственно вектор

$$\boldsymbol{e}_0 = \begin{pmatrix} \cos(\delta_0) \\ 0 \\ \sin(\delta_0) \end{pmatrix}$$

определяет точку на небесной сфере, в которую направлена оптическая ось камеры (α_0, δ_0). Векторы $\boldsymbol{F} = -F{\cdot}\boldsymbol{e}_0$ и $\boldsymbol{p} = -p{\cdot}\boldsymbol{e}$ описывают пути лучей света, идущих чрез объектив к центру пластинки и к точке P, в которой находится изображение звезды. Они образуют угол φ, так что

$$\cos(\varphi) = \boldsymbol{e}_0 \cdot \boldsymbol{e} = \cos(\delta_0)\cos(\delta)\cos(\alpha - \alpha_0) + \sin(\delta_0)\sin(\delta).$$



Комета Брэдфилда 4 сентября 1982 года

Звезды сравнения:
1 PPM 052974
2 PPM 052987
3 PPM 052990
4 PPM 053019
5 PPM 053028
6 PPM 053029
7 PPM 053050
8 PPM 053051
⊗ Комета

Координаты центра пластинки:
$\alpha \approx 12^h28^m48^s$
$\delta \approx 44°00'$

Рис. 12.1. Пример изображения звездного поля

В плоскости пластинки векторы

$$\boldsymbol{e}_X = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \quad \text{и} \quad \boldsymbol{e}_Y = \begin{pmatrix} +\sin(\delta_0) \\ 0 \\ -\cos(\delta_0) \end{pmatrix}$$

задают систему координат, ориентированную в направлении север–юг и восток–запад. Если координаты X и Y точки P выражены в единицах фокусного расстояния F, вектор $\boldsymbol{p}$ может быть представлен как

$$\boldsymbol{p} = \boldsymbol{F} + (F \cdot X)\cdot \boldsymbol{e}_X + (F \cdot Y)\cdot \boldsymbol{e}_Y.$$

Если переписать это уравнение в другом виде, соотношение между (α, δ) и (X, Y) можно представить следующим образом:

$$
\begin{aligned}
p\cos(\delta)\cos(\alpha-\alpha_0) &= F\cos(\delta_0) - FY\sin(\delta_0),\\
p\cos(\delta)\sin(\alpha-\alpha_0) &= -FX,\\
p\sin(\delta) &= F\sin(\delta_0) - FY\cos(\delta_0),
\end{aligned}
\qquad (12.1)
$$

где

$$p = |\boldsymbol{p}| = F\sqrt{1 + X^2 + Y^2}$$

или

$$p = F/\cos(\varphi) = F/(\cos(\delta_0)\cos(\delta)\cos(\alpha-\alpha_0) + \sin(\delta_0)\sin(\delta)).$$

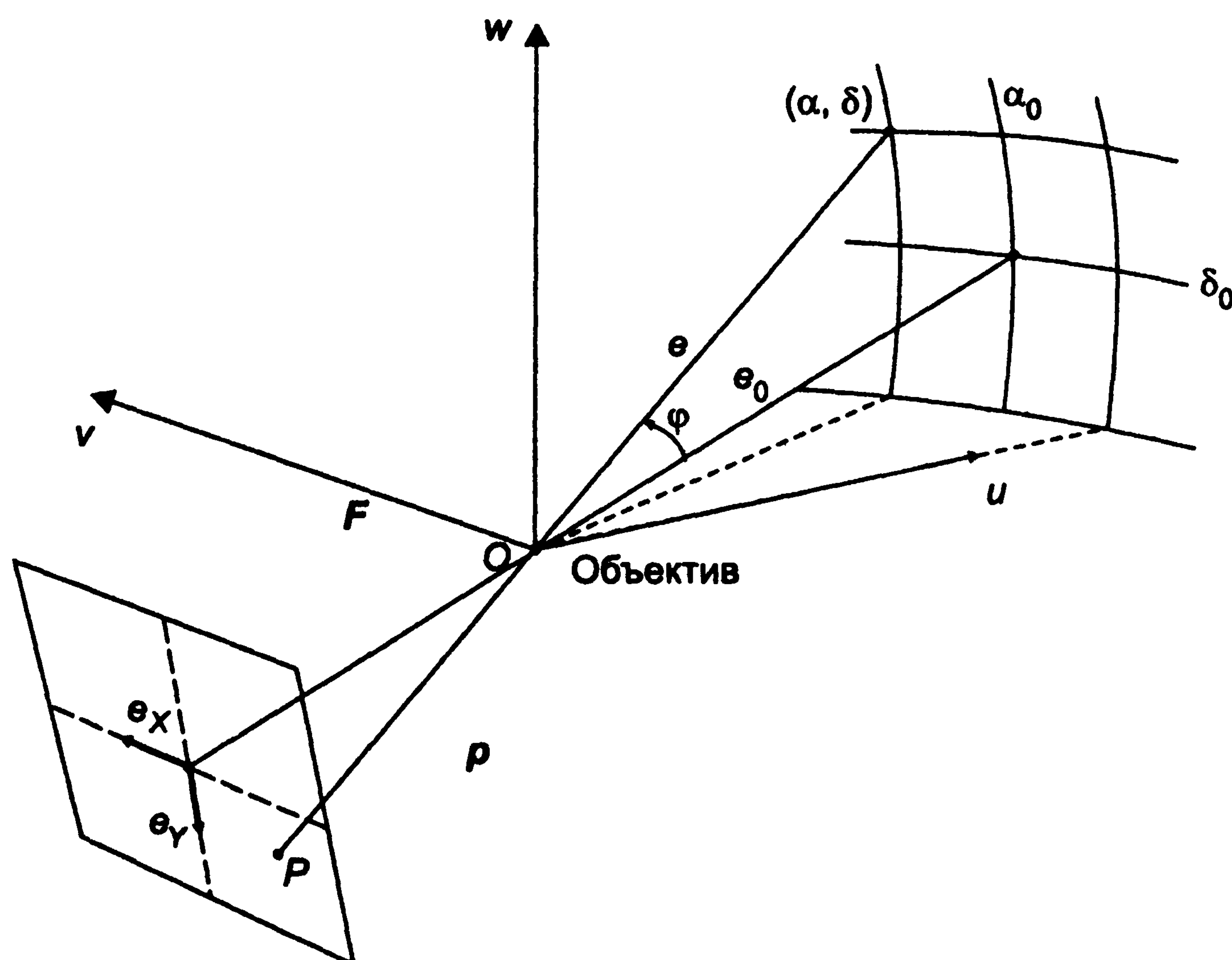


Рис. 12.2. Отображение звездного поля на фотопластинке

Решение для сферических координат дает следующие уравнения

$$
\begin{aligned}
\alpha &= \alpha_0 + \operatorname{arctg}\left\{\frac{-X}{\cos(\delta_0) - Y\sin(\delta_0)}\right\},\\
\delta &= \arcsin\left\{\frac{\sin(\delta_0) + Y\cos(\delta_0)}{\sqrt{1 + X^2 + Y^2}}\right\}.
\end{aligned}
\qquad (12.2)
$$

Обратное преобразование (12.2) легко может быть получено из (12.1), так что

$$
\begin{aligned}
X &= -\frac{\cos(\delta)\sin(\alpha-\alpha_0)}{\cos(\delta_0)\cos(\delta)\cos(\alpha-\alpha_0) + \sin(\delta_0)\sin(\delta)},\\
Y &= -\frac{\sin(\delta_0)\cos(\delta)\cos(\alpha-\alpha_0) - \cos(\delta_0)\sin(\delta)}{\cos(\delta_0)\cos(\delta)\cos(\alpha-\alpha_0) + \sin(\delta_0)\sin(\delta)}.
\end{aligned}
\qquad (12.3)
$$

Оба эти преобразования можно осуществить с помощью следующей функции.

```
//---------------------------------------------------------------------------
// StdEqu: Вычисление экваториальных координат по стандартным координатам
//
// Ввод:
//   RA0       Прямое восхождение оптической оси, рад
//   Dec0      Склонение оптической оси, рад
//   X         Стандартная координата X
//   Y         Стандартная координата Y
//
// Вывод:
//   RA        Прямое восхождение, рад
//   Dec       Склонение, рад
//
// Примечание: все углы задаются в радианах
//---------------------------------------------------------------------------
void StdEqu ( double RA0, double Dec0, double X, double Y,
              double& RA, double& Dec )
{
  RA  = RA0 + atan ( -X / (cos(Dec0)-Y*sin(Dec0)) );
  Dec = asin ( (sin(Dec0)+Y*cos(Dec0))/sqrt(1.0+X*X+Y*Y) );
}

//---------------------------------------------------------------------------
// EquStd: Вычисление стандартных координат по экваториальным
//
// Ввод:
//   RA0   Прямое восхождение оптической оси, рад
//   Dec0  Склонение оптической оси, рад
//   RA    Прямое восхождение, рад
//   Dec   Склонение, рад
//
// Вывод:
//   X   Стандартная координата X
//   Y   Стандартная координата Y
//---------------------------------------------------------------------------
void EquStd ( double RA0, double Dec0, double RA, double Dec,
              double& X, double& Y )
{
  //
  // Константы
  //
  const double c =
    cos(Dec0)*cos(Dec)*cos(RA-RA0)+sin(Dec0)*sin(Dec);

  X = - ( cos(Dec)*sin(RA-RA0) ) / c;
  Y = - ( sin(Dec0)*cos(Dec)*cos(RA-RA0)-cos(Dec0)*sin(Dec) ) / c;
}
```

12.2. Постоянные пластинки

Безразмерные координаты X и Y введены нами как стандартные[1] координаты, не зависящие от фокусного расстояния оптической системы, в системе коорди-

[1] В отечественной литературе встречается еще и другое их название — идеальные координаты. — *Примеч. перев.*

нат, связанной с направлением меридиана, проходящего через центр пластинки (α_0, δ_0). Если эта система координат используется для обработки пластинки, стандартные координаты получаются путем деления измеренных координат x и y на фокусное расстояние системы

$$X = x/F, \qquad Y = y/F.$$

Вообще, начало координат не обязательно точно совпадает с точкой пересечения оптической оси и плоскости пластинки. Чтобы учесть такое смещение (Δx, Δy), следует ввести в уравнения следующие поправки:

$$X = x/F - (\Delta x)/F, \qquad Y = y/F - (\Delta y)/F.$$

Если, кроме того, координатные оси повернуты относительно направления север–юг на угол γ, уравнения приобретают вид:

$$X = (x \cdot \cos(\gamma) - y \cdot \sin(\gamma))/F - (\Delta x)/F,$$
$$Y = (x \cdot \sin(\gamma) + y \cdot \cos(\gamma))/F - (\Delta x)/F.$$

Впрочем, смещение и вращение не единственные обстоятельства, нарушающие соответствие между измеренными и стандартными координатами. Аберрации оптической системы, а также наклон и искривление пленки требуют внесения дополнительных поправок. Таким образом, общий вид уравнений становится

$$X = a \cdot x + b \cdot y + c,$$
$$Y = d \cdot x + e \cdot y + f. \qquad (12.4)$$

Они содержат шесть величин a, b, c, d, e и f, именуемых *постоянными пластинки*. С их помощью возможно осуществить переход от координат (x, y) к стандартным координатам (X, Y). Постоянные пластинки, разумеется, не известны заранее и должны быть определены с помощью опорных звезд. Если нам известны экваториальные координаты $(\alpha_i, \delta_i)_{i=1,2,3}$ трех опорных звезд, то из выражения (12.3) мы получаем три уравнения

$$X_1 = x_1 \cdot a + y_1 \cdot b + c,$$
$$X_2 = x_2 \cdot a + y_2 \cdot b + c, \qquad (12.5)$$
$$X_3 = x_3 \cdot a + y_3 \cdot b + c,$$

которые могут быть решены для a, b и c. Оставшиеся постоянные пластинки d, e и f находим из соответствующих уравнений для Y_i. Знание фокусного расстояния в данном случае не требуется. Это важно, в частности, если мы обрабатываем увеличенное изображение, масштаб которого неизвестен.

Как мы уже убедились, постоянные пластинки можно найти, решив два уравнения, каждое из которых содержит три неизвестных. Для этого требуются по меньшей мере три опорных звезды с известными координатами. Поскольку измерение координат звезд не бывает совершенно свободно от ошибок, желательно использовать как можно больше опорных звезд. Обычно система уравнений вида

$$\begin{aligned} X_1 &= x_1 \cdot a + y_1 \cdot b + c \\ &\dots \\ X_n &= x_n \cdot a + y_n \cdot b + c \end{aligned} \tag{12.6}$$

при $n > 3$ не может быть решена. Обойти эту трудность позволяет метод наименьших квадратов, который мы сейчас опишем.

12.3. Метод наименьших квадратов

Вообще, если мы хотим определить m неизвестных $\boldsymbol{x} = (x_1, ..., x_m)$ из n уравнений

$$\boldsymbol{a}_i \cdot \boldsymbol{x} = a_{i1}x_1 + \ldots + a_{im}x_m = b_i \quad (i = 1, \ldots, n) \tag{12.7}$$

с заданными коэффициентами x_i и a_{ij}, тогда при $n > m$ мы не можем ожидать, что все уравнения будут решены одновременно[1]. Сперва придется ввести дополнительные члены, известные под названием *остатков.*

$$\nu_i = a_{i1} \cdot x_1 + \ldots + a_{im}x_m - b_i \quad (i = 1, \ldots, n). \tag{12.8}$$

Вместо попыток найти значения $x_1, ..., x_m$, при которых i обращаются в нуль (что соответствует решению всех уравнений), мы будем искать наименьшее значение суммы квадратов остатков

$$\chi = \sum_{i=1}^{n} \nu_i^2 .$$

Для этого вначале преобразуем уравнения (12.8) в эквивалентную систему, имеющую вид:

$$\begin{array}{lll} \mu_1 & = & r_{11} \cdot x_1 + r_{12} \cdot x_2 + \ldots + r_{1m} \cdot x_m - d_1 \\ \mu_2 & = & \qquad\qquad r_{22} \cdot x_2 + \ldots + r_{2m} \cdot x_m - d_2 \\ \vdots & = & \qquad\qquad\qquad \ddots \qquad \vdots \qquad\quad \vdots \\ \mu_m & = & \qquad\qquad\qquad\qquad r_{mm} \cdot x_m - d_m \\ \mu_{m+1} & = & \qquad\qquad\qquad\qquad\qquad\quad - d_{m+1} \\ \vdots & = & \qquad\qquad\qquad\qquad\qquad\quad \vdots \\ \mu_n & = & \qquad\qquad\qquad\qquad\qquad\quad - d_n , \end{array} \tag{12.9}$$

Здесь все коэффициенты r_{ij} равны нулю при $i > j$, что существенно упрощает решение системы.

---

[1] В этом разделе использованы некоторые символы, вводившиеся ранее для обозначения других величин. В частности, обозначения a, b и x использованы при изложении метода наименьших квадратов для обозначения коэффициентов и неизвестных в системе уравнений, хотя мы использовали их для обозначения постоянных пластинки и координат звезд.

Пусть c и s два действительных числа, таких что $c_2 + s_2 = 1$, тогда сумма квадратов остатков не изменится в случае замены, например, v_1 на $v'_1 = p \cdot v_1 - q \cdot v_2$ и v_2 на $v'_2 = q \cdot v_1 - p \cdot v_2$, поскольку ввиду сделанного нами предположения

$$v'^2_1 + v'^2_2 = c^2 v_1^2 - 2csv_1v_2 + s^2v_2^2 + s^2v_1^2 + 2csv_1v_2 + c^2v_2^2 = \\ = (c^2 + s^2) \cdot (v_1^2 + v_2^2)$$

эквивалентно $v_1^2 + v_2^2$. Если подобрать числа c и s так, что

$$c = +a_{11}/h$$

и

$$s = +a_{21}/h \quad \text{при} \quad h = \pm\sqrt{a_{11}^2 + a_{11}^2},$$

тогда, как нам и требуется, a_{21} заменяется на

$$a'_{21} = -sa_{11} + ca_{21} = (-a_{21}a_{11} + a_{11}a_{21})/h = 0.$$

С помощью такой перегруппировки (известной под названием поворота Гивенса) так же определяются коэффициенты $a_{31} \dots a_{n1}$, $a_{32} \dots a_{n2}$, ..., и $a_{m+1,m} \dots a_{nm}$, в результате чего получается система уравнений вида (12.9).

Если в изначальной системе $\boldsymbol{Ax} = \boldsymbol{b}$ матрица коэффициентов $\boldsymbol{A} = (a_{ij})_{i=1\dots n, j=1\dots m}$ заполнена полностью, то преобразованная система

$$\begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{pmatrix} \boldsymbol{x} = \boldsymbol{d} \qquad (12.10)$$

имеет треугольную матрицу, в которой элементы ниже диагонали отсутствуют.

Теперь сумма квадратов остатков может быть записана в виде выражения, состоящего из двух членов

$$\chi = |v|^2 = |\mu|^2 = \sum_{i=1}^{m} \mu_i^2 + \sum_{i=m+1}^{m} d_i^2$$

из которых только первый остается зависим от неизвестной величины x_k. Вторая сумма, очевидно, может быть получена в том случае, когда все значения μ_i равны нулю при $i = 1 \dots m$. Соответствующие значения x_k получаются путем обратной подстановки:

$$x_m = d_m / r_{mm}, \\ x_k = (d_k - \sum_{l=k+1}^{m} r_{kl}x_l)/r_{kk} \qquad (k = m-1, \dots, 1). \qquad (12.11)$$

Описанное преобразование системы уравнéний в треугольную форму и последующее ее решение осуществляется с помощью класса `solverLSQ`, обеспечивающего простой способ решения линейных уравнений подобного рода.

```
// Класс SolverLSQ для метода наименьших квадратов
//
class SolverLSQ
{
  public:
    // Конструктор, деструктор
    SolverLSQ(int n); // n: Число определяемых параметров
    ~SolverLSQ();

    // Функции-члены

    // Переход к следующей задаче
    void Init();

    // Подстановка Ax=b
    // (перегруппировка с использованием поворота Гивенса)
    void Accumulate (const double A[], double b);

    // Решение задачи для вектора x[] путем обратной подстановки.
    void Solve (double x[]);

  private:
    int       N;         // Число определяемых параметров
    double    *d;        // Правая часть видоизмененных уравнений
    double    **R;       // Верхняя правая часть треугольной матрицы

};
```

Для решения системы уравнений вида $Ax = b$ с n неизвестными следует объявить объект класса solverLSQ. Используя известную размерность, конструктор зарезервирует и инициализирует динамический массив для хранения правой верхней треугольной матрицы R и вектора d преобразованной системы.

```
// Constructor; n: Число определяемых параметров
//
SolverLSQ::SolverLSQ (int n)
  : N(n)
{
  // Выделение памяти для R и ж
  d = new double[N];
  R = new double*[N];

  for (int i=0; i<N; i++) R[i] = new double[N];

  Init(); // Новый цикл для d и R
}
//
// Переход к следующей задаче
//
void SolverLSQ::Init()
{
  // Новый цикл для d и R
  for (int i=0; i<N; i++) {
    d[i]=0.0;
    for (int j=0;j<N;j++) R[i][j]=0.0;
  }
}
```

Для каждой строки $a_i \cdot x = b_i$ ($i = 1, ..., n$) системы уравнений с помощью метода Accumulate коэффициенты a_i и b_i будут последовательно преобразовываться в описанную выше треугольную матрицу $\boldsymbol{R}$ и вектор $\boldsymbol{d}$.

```
// Подстановка Ax=b
// (перегруппировка с использованием поворота Гивенса)
//
void SolverLSQ::Accumulate (const double A[], double b)
{
  // Переменные
  //
  int      i,j;
  double   c,s,h;
  double*  a = new double[N];

  for (i=0;i<N;i++) a[i]=A[i];   // Copy A

  // Выполнение поворота Гивенса
  for (i=0; i<N; i++)
  {
    // Выполнение поворота Гивенса для определения i-го элемента a
    if ( R[i][i]==0.0 &&  a[i]==0.0 ) {
      c = 1.0; s = 0.0; R[i][i] = 0.0;
    }
    else {
      h = sqrt ( R[i][i]*R[i][i] + a[i]*a[i] );
      if (R[i][i]<0.0) h=-h;
      c = R[i][i]/h;
      s = a[i]/h;
      R[i][i] = h;
    };

    a[i] = 0.0;

    // Выполнение поворота для оставшихся элементов a
    for (j=i+1; j<N; j++) {
      h       = +c*R[i][j]+s*a[j];
      a[j]    = -s*R[i][j]+c*a[j];
      R[i][j] = h;
    }

    // Выполнение поворота для i-го элемента d
    h    = +c*d[i]+s*b;
    b    = -s*d[i]+c*b;
    d[i] = h;
  }

  delete[] a;
}
```

С помощью метода Solve мы, наконец, получим после обратной подстановки искомое решение системы уравнений.

```
// Решение задачи для вектора x[] путем обратной подстановки
//
void SolverLSQ::Solve (double x[])
{
```

```
  // Переменные
  //
  int i,j; i=j=0;
  double Sum=0.0;

  // Проверка матрицы на сингулярность
  for (i=0;i<N;i++)
    if ( R[i][i] == 0.0 ) {
      cerr << " ERROR: Singular matrix R in SolverLSQ::Solve()" << endl;
      exit(1);
    };

  // Решение Rx=d для x_n,...,x_1 путем обратной подстановки
  x[N-1] = d[N-1] / R[N-1][N-1];
  for (i=N-2;i>=0;i--) {
    Sum = 0.0;
    for (j=i+1;j<N;j++) Sum += R[i][j]*x[j];
    x[i] = ( d[i] - Sum ) / R[i][i];
  };
}
```

Здесь, как принято в языке C++, индексы массива пробегают значения от 0 до $n - 1$.

12.4. Программа Foto

Программа Foto позволяет получить точные координаты звезд, комет или астероидов по фотографическим изображениям. Она освобождает пользователя от трудностей, связанных с вычислением стандартных координат и получением решений для остатков.

Прежде чем обратиться к программе, следует подготовить необходимые данные ввода. Вначале, с помощью звездной карты следует отождествить хотя бы несколько ярчайших звезд, которые позволят определить приблизительные координаты области неба, отображенной на пластинке. После этого можно выбрать несколько звезд, координаты которых берутся из звездного каталога, как, например, каталог PPM. При необходимости учитываются собственные движения звезд в период между каталожной эпохой и датой получения фотографии. Опорные звезды должны быть равномерно распределены по пластинке, а их изображения должны иметь минимальные размеры. Это позволит повысить точность измерений. Координаты опорных звезд и исследуемого объекта без труда могут быть найдены путем наложения на увеличенное изображение прозрачной миллиметровой сетки, ориентированной приблизительно в направлении север–юг. Наконец, мы определяем прямое восхождение и склонение центра пластинки, которые необходимы для вычисления стандартных координат. Поскольку координаты центра пластинки оказывают незначительное влияние на определение положения объекта, достаточно определить их приблизительно с помощью звездного атласа.

Все данные заносятся в файл **Foto.dat**. Его первая строка содержит координаты — прямое восхождение (^{h}, ^{m}, ^{s}) и склонение (°, ′, ″) центра пластинки. Затем

указываются координаты отдельных объектов на пластинке. Опорные звезды, используемые для определения постоянных пластинки, отмечены звездочкой (*) в первом столбце. После наименования звезды, которое может содержать до 11 знаков, указываются измеренные экваториальные координаты (x, y в миллиметрах), а также экваториальные координаты опорных звезд (^{h}, ^{m}, ^{s} и °, ′, ″). Ниже приводится пример данных, относящихся к пластинке, схематически изображенной на рис. 12.1.

```
CENTRE                      12 28 45.000  +44 00 00.00
*PPM 052974   +47.5  +73.3  12 25 06.952  +44 47 55.79
*PPM 052987   +41.2  -62.1  12 25 59.995  +43 07 05.93
*PPM 052990   +29.9  +65.2  12 26 22.451  +44 42 39.02
*PPM 053019    -4.8  -72.7  12 29 09.766  +43 00 54.07
*PPM 053028   -33.0  +21.0  12 30 52.966  +44 11 19.19
*PPM 053029   -31.2  -62.0  12 30 53.525  +43 09 30.08
*PPM 053050   -51.2  -59.7  12 32 17.476  +43 11 57.76
*PPM 053051   -55.7  +60.6  12 32 21.122  +44 40 54.24
BRADFIELD     +29.2  -42.1
```

Функция Getinp считывает и размещает надлежащим образом эти данные. Затем находятся стандартные координаты опорных звезд, знание которых требуется для определения постоянных пластинки. После этого процедура может быть обращена: стандартные и экваториальные координаты всех объектов на пластинке могут быть получены из измеренных координат. Сравнив их с известными координатами опорных звезд, теперь можно оценить ошибки, возникающие при обработке пластинки. Существенные ошибки, такие как ошибки в отождествлении звезд, легко распознаются.

```
//------------------------------------------------------------------------
// Модуль:    Программа Foto (Foto.cpp)
// Задача:    Астрометрическая обработка фотопластинок
// (c) 1999 Oliver Montenbruck, Thomas Pfleger
//------------------------------------------------------------------------

#include <cmath>
#include <fstream>
#include <iomanip>
#include <iostream>

#include "APC_Const.h"
#include "APC_IO.h"
#include "APC_Math.h"
#include "APC_Spheric.h"

#ifdef __GNUC__  // адаптация для GNU C++
#include "GNU_iomanip.h"
#endif

using namespace std;

//
// Константы
//
const int MaxObj = 30;    // Наибольшее число объектов на фотографии
```

```
const int StrLen = 13;
//------------------------------------------------------------------------
// Чтение входных данных из файла
//
// Ввод:
// Имя входного файла
//
// Вывод:
//   RA0       Прямое восхождение центра пластинки
//   Dec0      Склонение центра пластинки
//   NObj      Число объектов
//   Name[]    Названия объектов
//   RA[]      Прямое восхождение опорных объектов
//   Dec[]     Склонение опорных объектов
//   x[]       Измеренные координаты опорных объектов
//   y[]       относительно центра пластинки, мм
//
// Примечание: все углы задаются в радианах
//------------------------------------------------------------------------
void GetInput ( char* Filename,
                double& RA0, double& Dec0, int& NObj, char Name[][StrLen],
                double RA[], double Dec[], double x[], double y[] )
{
  //
  // Переменные
  //
  int      i,k,h,m,deg,min;
  double   s,sec;
  char     c,sign;
  ifstream inp;

  // Открытие файла для чтения
  inp.open(Filename);
  cout << " Input data file: " << Filename << endl << endl;

  // Чтение координат центра пластинки
  for (k=0;k<StrLen;k++) inp.get(c);

  inp >> h >> m >> s;       inp.get(c).get(c).get(sign);
  inp >> deg >> min >> sec; inp.ignore(81,'\n');

  RA0  = Rad*15.0*Ddd(h,m,s);
  Dec0 = Rad*Ddd(deg,min,sec); if (sign=='-') Dec0 = -Dec0;

  // Чтение названия объекта, измеренных и экваториальных координат
  i = -1;
  do {
    ++i;
    inp.get(Name[i],StrLen);  if (inp.fail()) break;
    if ( Name[i][0] == '*' ) {

      // Опорная звезда
      inp >> x[i] >> y[i];
      inp >> h >> m >> s;       inp.get(c).get(c).get(sign);
      inp >> deg >> min >> sec; inp.ignore(81,'\n');

      RA[i]  = Rad*15.0*Ddd(h,m,s);
```

```
      Dec[i] = Rad*Ddd(deg,min,sec); if (sign=='-') Dec[i] = -Dec[i];
    }
    else {

      // Исследуемый объект
      inp >> x[i] >> y[i]; inp.ignore(81,'\n');
      RA[i]=0.0; Dec[i]=0.0;
    };

    NObj = i+1;
    c=inp.peek(); // Проверка достижения конца файла
  }
  while ( !inp.eof() && i<MaxObj );

  inp.close();  // Close input file
}


//------------------------------------------------------------------------------
//
// Основная программа
//
//------------------------------------------------------------------------------
void main(int argc, char* argv[])
{
  //
  // Константы
  //
  const int NEst = 3;   // Число определяемых параметров

  //
  // Переменные
  //
  int        i, NObj;
  char       Name[MaxObj][StrLen];
  double     RA0, Dec0, RA_Obs, Dec_Obs, D_RA, D_Dec;
  double     RA[MaxObj], Dec[MaxObj], Delta[MaxObj];
  double     x[MaxObj], y[MaxObj],
  double     X[MaxObj], Y[MaxObj];
  double     A[NEst],C[NEst];
  double     a, b, c, d, e, f;
  double     Det, FocLen, Scale;
  SolverLSQ LSQx(NEst);
  SolverLSQ LSQy(NEst);
  char       InputFile[APC_MaxFilename] = "";
  char       OutputFile[APC_MaxFilename] = "";
  bool       FoundInputfile = false;
  bool       FoundOutputfile = false;
  ofstream   OutFile;

  // Название
  cout << endl
       << "     FOTO: astrometric analysis of photographic plates   " << endl
       << "           (c) 1999 Oliver Montenbruck, Thomas Pfleger   " << endl
       << endl;
```

```
// Поиск входного файла
GetFilenames( argc, argv, "Foto.dat", InputFile, FoundInputfile,
              OutputFile, FoundOutputfile );

// Прерывание программы, если входной файл не найден
if (!FoundInputfile) {
  cerr << " Terminating program." << endl;
  exit(-1);
}

// Чтение файла ввода
GetInput (InputFile, RA0,Dec0, NObj, Name, RA,Dec, x,y);

// Вычисление стандартных
// координат опорных звезд и заполнение остальных квадратов
for (i=0;i<NObj;i++)
  if (Name[i][0]=='*') {
    EquStd (RA0, Dec0, RA[i], Dec[i], X[i], Y[i]);
    A[0]=x[i]; A[1]=y[i]; A[2]=1.0;
    LSQx.Accumulate (A, X[i]);
    LSQy.Accumulate (A, Y[i]);
  };

// Вычисление постоянных пластинки
LSQx.Solve(C); a=C[0]; b=C[1]; c=C[2];
LSQy.Solve(C); d=C[0]; e=C[1]; f=C[2];

// Вычисление экваториальных координат (и ошибок по опорным звездам () )
for (i=0;i<NObj;i++) {
  X[i] = a*x[i]+b*y[i]+c;
  Y[i] = d*x[i]+e*y[i]+f;
  StdEqu (RA0, Dec0, X[i], Y[i], RA_Obs, Dec_Obs);
  if (Name[i][0]=='*') {
    D_RA  = (RA_Obs-RA[i]) * cos(Dec[i]);
    D_Dec = (Dec_Obs-Dec[i]);
    Delta[i] = Arcs * sqrt(D_RA*D_RA+D_Dec*D_Dec);  // in [arcsec] - в угловых сек.
  };
  RA[i] = RA_Obs;  Dec[i] = Dec_Obs;
};

// Вычисление фокусного расстояния и масштаба
Det    = a*e-d*b;
FocLen = 1.0/sqrt(fabs(Det)); // in [mm] - в мм
Scale  = Arcs / FocLen;       // in ["/mm] - в "/мм

// Переадресация выходных данных в случае, если должен быть создан файл вывода
if (FoundOutputfile) {
  OutFile.open(OutputFile);
  if (OutFile.is_open())
    cout = OutFile;
}

// Вывод
cout << " Plate constants:" << endl << endl
     << fixed << setprecision(8)
     << "   a =" << setw(12) << a << "   b =" << setw(12) << b
     << "   c =" << setw(12) << c << endl
```

```
      << "   d =" << setw(12) << d << "   e =" << setw(12) << e
      << "   f =" << setw(12) << f << endl << endl;

  cout << " Effective focal length and image scale:" << endl << endl
      << fixed << setprecision(2)
      << "   F =" << setw(9) << FocLen << " mm" << endl
      << "   m =" << setw(7) << Scale  << " \"/mm" << endl << endl;

  cout << " Coordinates:" << endl << endl
      << "         Name         x     y       X       Y "
      << "          RA            Dec    Error" << endl
      << "                     mm    mm                   "
      << "      h  m  s       o  '  \"      \" " << endl;

  for (i=0;i<NObj;i++) {
    cout << "    " << Name[i] << fixed
         << setprecision(1) << setw(7) << x[i] << setw(7) << y[i]
         << setprecision(4) << setw(9) << X[i] << setw(8) << Y[i]
         << setprecision(2) << setw(14) << Angle(Deg*RA[i]/15,DMMSSs)
         << showpos << "  "
         << setprecision(1) << setw(11) << Angle(Deg*Dec[i],DMMSSs)
         << noshowpos;
    if (Name[i][0]=='*') cout << setprecision(1) << setw(6) << Delta[i];
    cout << endl;
  }

  if (OutFile.is_open()) OutFile.close();
}
```

На основании приведенных выше данных, относящихся к пластинке с изображением кометы Бредфилда, программа Foto выдает следующую распечатку:

```
     FOTO: astrometric analysis of photographic plates
          (c) 1999 Oliver Montenbruck, Thomas Pfleger

 Using default input file Foto.dat
 Input data file: Foto.dat

 Plate constants:

   a =  0.00021651   b =  0.00000827   c =  0.00035472
   d = -0.00000799   e =  0.00021639   f = -0.00149942

 Effective focal length and image scale:

   F =  4616.79 mm
   m =  44.68 "/mm

 Coordinates:

        Name        x      y       X       Y          RA           Dec     Error
                   mm     mm                       h  m  s       o  '  "      "
   *PPM 052974    47.5   73.3   0.0112  0.0140   12 25 07.11   +44 47 50.9   5.2
   *PPM 052987    41.2  -62.1   0.0088 -0.0153   12 25 59.96   +43 07 23.9  18.0
   *PPM 052990    29.9   65.2   0.0074  0.0124   12 26 22.46   +44 42 25.9  13.2
   *PPM 053019    -4.8  -72.7  -0.0013 -0.0172   12 29 09.18   +43 00 54.0   6.4
   *PPM 053028   -33.0   21.0  -0.0066  0.0033   12 30 51.88   +44 11 18.0  11.7
   *PPM 053029   -31.2  -62.0  -0.0069 -0.0147   12 30 55.30   +43 09 30.5  19.5
```

*PPM 053050	-51.2	-59.7	-0.0112	-0.0140	12 32 16.68	+43 11 38.5	21.1
*PPM 053051	-55.7	60.6	-0.0112	0.0121	12 32 21.67	+44 41 14.4	21.0
BRADFIELD	29.2	-42.1	0.0063	-0.0108	12 26 45.28	+43 22 39.7	

После определения постоянных пластинки находим эффективное фокусное расстояние системы (фокусное расстояние камеры, умноженное на примененное увеличение при печати фотографии) и масштаб изображения. После этого измеренные координаты каждого объекта (x, y) переводятся в стандартные координаты (X, Y), и определяются его экваториальные координаты. Опорные звезды по измерениям снимков показывают уклонения от каталожных координат в пределах 10″–20″, что соответствует обычным ошибкам в 1/4–1/2 мм при использовании миллиметровой сетки. Более точные результаты получаются при обработке изображений на специальном измерительном устройстве.

Собственные движения звезд в промежутке между эпохой каталога PPM (2000.0) и моментом наблюдения (1982.8) не учитываются, поскольку для большинства опорных звезд они составляют величину около 2″, что значительно меньше величины ошибки при определении координат описанным способом.

12.5. Каталог PPM (Position and Proper Motion Catalogue)

Среди многих доступных каталогов звезд выделяется каталог PPM, составленный С. Рёзером и У. Бастианом, который обеспечивает возможность астрометрической обработки фотографий звездных полей благодаря своей точности и количеству охваченных звезд. Его стоит использовать во всех случаях, когда требуется точность около 1″ или когда предельная звездная величина на снимке составляет 11^m. В других случаях, когда требования более жесткие, можно воспользоваться каталогами вроде Hipparcos Star Catalogue или Hubble Guide Star Catalogue. Благодаря охвату слабых звезд эти каталоги позволяют производить измерения даже весьма ограниченных по полю изображений, полученных с помощью ПЗС камер.

По соглашению между авторами настоящей книги и Государственным астрономическим институтом в Гайдельберге (Astronomisches Recheninstitut Heidelberg) полная версия каталога PPM, включая соответствующие комментарии, представлена на прилагаемом компакт-диске. Каталог изначально состоял из четырех частей, которые объединены в единую базу данных PPM.dat:

- PPM для северной полусферы (№ 1–181731).
- PPM для южной полусферы (№ 181732–378910).
- PPM, дополнение, содержащее координаты 275 ярких звезд (№ 400001–400321).
- PPM, дополнение, содержащее координаты 90 000 южных звезд (№ 700001–789676).

Файл содержит сведения о 468 861 звездах и состоит из соответствующего числа строк по 131 символ каждая. В табл. 12.1 расшифрованы сведения, содержащиеся в этих строках.

Таблица 12.1. Структура каталога PPM

Столбцы	Содержание
2– 7	Номер по каталогу PPM
10–18	Номер по Боннскому/Кордовскому Обозрению
20–23	Звездная величина
25–26	Спектральный класс
28–29	Прямое восхождение J2000 (часы)
31–32	Прямое восхождение J2000 (минуты)
34–39	Прямое восхождение J2000 (секунды)
42	Склонение J2000 (знак)
43–44	Склонение J2000 (градусы)
46–47	Склонение J2000 (минуты)
49–53	Склонение J2000 (секунды)
56–62	Собственное движение по прямому восхождению (секунды за столетие)
64–69	Собственное движение по склонению (секунды за столетие)

Хотя файл содержит исключительно печатаемые символы ASCII и в принципе может быть прочитан в текстовом редакторе, эта процедура заняла бы слишком много времени ввиду его размера, составляющего около 60 Мбайт. Для облегчения этой задачи на компакт-диске представлены две вспомогательные программы PPMbin и PPMcat.

С помощью программы PPMbin наиболее важные сведения (номер, координаты, собственное движение и звездная величина) считываются из каталога и размещаются в двоичном файле **PPM.bin**:

```
C> PPMbin D:/APCe/Data/PPM/PPM.dat PPMbin
PPMbin: Conversion of the PPM star catalogue (ASCII->binary)
(c) 1999 Oliver Montenbruck, Thomas Pfleger
***********************************************
468861 stars processed
```

Эта процедура требует меньше места на диске и выполняется относительно быстро. Во время выполнения программы после обработки очередных 10 000 строк данных на экране появляется звездочка (*), и на медленно работающем компьютере можно следить за ее работой. Обычно это занимает несколько минут. Для облегчения задачи специально разработанные как для Windows, так и для Linux версии программы **PPM.bin** представлены на компакт-диске.

Наконец, программа PPMcat позволяет отобрать звезды, содержащиеся в каталоге PPM и попадающие в поле фотографического изображения.

```
C> PPMcat D:/APC/Data/PPM/Win32/PPM.bin
      PPMcat: Selection of stars from the PPM Catalogue
          (c) 1999 Oliver Montenbruck, Thomas Pfleger

 Image centre
   Right ascension [hh mm.m] ...   12 28.75
   Declination [deg]         ...   44.0
```

```
Radius [deg]                  ...   1.0
Epoch [yyyy.y]                ...   1982.7

PPM Number        RA            Dec          PM(RA)  PM(Dec)   m
              h  m  s         o  '  "       s/cy    "/cy     mag
PPM  52981  12 25 45.107  +44 44 03.74  -0.730   -1.90    8.2
PPM  52983  12 25 53.345  +44 37 34.07  +0.120   -3.60   11.6
...
PPM  53039  12 31 39.100  +44 04 05.16  -0.140   -0.30   10.9
PPM  53044  12 31 41.746  +43 13 06.26  -0.440   -1.90   10.6
PPM  53046  12 31 59.476  +43 28 58.62  -0.260   +0.10   10.6
PPM  53051  12 32 21.186  +44 40 54.45  -0.370   -1.20    9.7
PPM  53062  12 33 39.312  +44 05 47.71  -1.300   -6.90    7.9
PPM  53065  12 34 00.512  +44 09 06.19  -0.220   -1.50   10.2
PPM 400176  12 28 04.711  +44 47 39.35  -1.800   -2.00    7.5
```

Вывод данных производится в формате, позволяющем использовать их в программе Foto. Для этих целей можно вводить имя файла вывода во время работы программы PPMcat (см. П.1.3). Исходный код PPMcat также находится на прилагаемом к книге компакт-диске.

Приложение

П.1. Прилагаемый компакт-диск

П.1.1. Содержание

На прилагаемом компакт-диске содержится полный комплект исходных текстов описанных в книге программ, а также файлы с необходимыми для них входными данными. Кроме того, на диске размещены исполняемые модули программ для компьютеров, совместимых с IBM PC, которые можно запускать в операционных системах Windows 95/98/NT и Linux. В таблице приводится сводка файлов, содержащихся в различных отдельных каталогах (папках) диска. Буква, назначенная диску (у нас D:), может, конечно, меняться в зависимости от конфигурации компьютера.

Папка/Каталог	Описание
D:\APCe\Win32\	Исполняемые программы (32-битные версии для Windows) AOEcat.exe, Coco.exe, Comet.exe, Eclipse.exe, EclTimer.exe, Foto.exe, Gauss.exe, Luna.exe, Numint.exe, Occult.exe, Phases.exe, Phys.exe, Planpos.exe, Planrise.exe, PPMbin.exe, PPMcat.exe, Sunset.exe
\Linux\	Исполняемые программы (версии для Linux) AOEcat, Coco, Comet, Eclipse, EclTimer, Foto, Gauss, Luna, Numint, Occult, Phases, Phys, Planpos, Planrise, PPMbin, PPMcat, Sunset UNIX-архивы файлов данных, исполняемых программ (только для Linux) и исходных текстов apcedat.tar, apceexe.tar, apcesrc.tar (включая make-файл)
\Source\	Исходные тексты библиотечных модулей AOEcat.cpp, APC_Cheb.cpp, APC_Cheb h, APC_Const.h, APC_DE.cpp, APC_DE.h, APC_IO.cpp, APC_IO.h, APC_Kepler.cpp, APC_Kepler.h, APC_Math.cpp, APC_Math h, APC_Moon.cpp, APC_Moon.h, APC_Phys cpp, APC_Phys.h, APC_Planets.cpp, APC_Planets.h, APC_PrecNut.cpp, APC_PrecNut.h, APC_Spheric.cpp, APC_Spheric.h, APC_Sun.cpp, APC_Sun h, APC_Time.cpp, APC_Time h, APC_VecMat3D.cpp, APC_VecMat3D.h, Coco cpp, Comet.cpp, Eclipse.cpp, EclTimer.cpp, Foto.cpp, Gauss.cpp, GNU_iomanip.h, Luna.cpp, Numint cpp, Occult.cpp, Phases cpp, Phys.cpp, Planpos.cpp, Planrise.cpp, PPMbin.cpp, PPMcat.cpp. Sunset.cpp
D:\APCe\Data\	Файлы с примерами исходных данных Comet.dat, Foto.dat, Gauss.dat, Numint.dat, Occult.dat, Orkisz.dat, ZC.dat База данных по элементам орбит астероидов и документация к ней \Aoe\astorb.dat, \Aoe\astorb html Каталог положений и собственных движений звезд \Ppm\PPM.dat, \Ppm\PPMbss.txt, \Ppm\PPMnorth.txt, \Ppm\PPMsouth.txt, \Ppm\PPMsupp txt, \Ppm\ReadMe.txt, \Ppm\Win32\PPM bin, \Ppm\Linux\PPM bin

Программы \Win32*.exe и двоичный файл \Data\Ppm\Win32\Ppm.bin предназначены исключительно для использования в среде Microsoft Windows (32-битная версия для процессоров Intel). Аналогично программы из каталога Linux и двоичный файл \Data\Ppm\Linux\PPM.bin могут использоваться только на ПК Intel x86, работающих под управлением Linux. Файлы с расширениями *.h, *.cpp и *.dat содержат тексты в формате ASCII и могут читаться и обрабатываться на компьютерах любой архитектуры и под любой операционной системой (UNIX, Macintosh, VMS). При этом надо учитывать, что имена каталогов могут в этом случае отличаться от указанных в соответствии с правилами той или иной операционной системы.

П.1.2. Требования к системе

Для достижения приемлемой производительности прилагаемых программ рекомендуются следующие характеристики персонального компьютера.

- Процессор Intel Pentium 133 МГц.
- Оперативная память 16–32 Мбайт.
- Свободное пространство на жестком диске 10–150 Мбайт.
- Привод CD-ROM.
- Операционная система Windows 95/98/NT или SuSE Linux 6.2.
- Компилятор Microsoft Visual C++ Version 5.0 или GNU C++ 2.91 (не обязательно).

Тем пользователям, кто намерен самостоятельно компилировать исходные тексты или использовать их для создания собственных программ, рекомендуется воспользоваться компьютером большей производительности. В частности, среда разработки компилятора C++ предъявляет более высокие требования к объему оперативной памяти и дисковому пространству.

Программы могут выполняться и на более слабых компьютерах (Intel 486, 50 МГц, 8 Мбайт). Однако принципиальным моментом является установка 32-разрядной операционной системы (Windows или Linux). В среде DOS (16-разрядной) или Windows 3.1 размещенные на диске программы (*.exe) работать не могут. При недостатке свободного места на жестком диске программы можно запускать непосредственно с компакт-диска.

Для модификации программ и разработки собственных приложений можно использовать различные компиляторы и отладочные среды. Однако особенности их реализации могут потребовать внесения небольших поправок в исходные тексты. Для разработки программ использовался компилятор Visual C++ 5.0, и в случае его применения для компиляции программ в них не требуется вносить никаких изменений.

Исходные тексты программ и файлы данных можно использовать не только в среде Windows и Linux, но и на других компьютерах, поддерживающих формат ISO 9660. В виду большого разнообразия существующих систем, мы не можем привести здесь какие-либо общие рекомендации относительно подходящих ком-

пиляторов и сред разработки. В определенных случаях может потребоваться внесение изменений в исходные тексты программ.

Имена файлов на компакт-диске могут некорректно воспроизводиться некоторыми операционными системами (например, они могут переводиться в верхний регистр). Поэтому рекомендуется (особенно для операционных систем семейства UNIX) извлекать файлы с исходными текстами из tar-архивов, содержащихся в каталоге Linux. В этом случае обеспечивается корректное воспроизведение имен файлов.

П.1.3. Запуск программ

Для запуска программ на компьютере под управлением операционной системы Windows файлы из папки D:\APCe\Win32 на компакт-диске надо скопировать в каталог на жестком диске (например, C:\APCe\Win32). Входные данные из папки D:\APCe\Data следует скопировать в ту же папку. Затем надо переключиться в режим командной строки DOS, запустив command.com в Windows 95/98 или cmd.exe в Windows NT, и перейти в соответствующую папку. После этого отдельные программы можно запускать следующим образом:

```
C:\>cd C:\APCe\Exe
C:\APCe\Exe>Sunset

     SUNSET: solar and lunar rising and setting times
       (c) 1999 Oliver Montenbruck, Thomas Pfleger

 First date (yyyy mm dd)              ...
```

Для многих программ удобно (а для некоторых необходимо) указывать в командной строке входные и выходные файлы (табл. П.1).

Таблица П.1. Вызов программ с передачей параметров в командной строке

Имя	Параметры	Исходные данные по умолчанию	Результаты по умолчанию
AOEcat	[<in-file> [out-file]]	astorb.dat	На экране
Comet	[<in-file> [out-file]]	Comet.dat	На экране
Eclipse	[out-file]		На экране
Foto	[<in-file> [out-file]]	Foto.dat	На экране
Gauss	[<in-file> [out-file]]	Gauss.dat	На экране
Numint	[<in-file> [out-file]]	Numint.dat	На экране
Occult	[<in-file> [out-file]]	Occult.dat	На экране
PPMbin	[<in-file> [out-file]]	PPM.dat	PPM.bin
PPMcat	[<in-file> [out-file]]	PPM.bin	На экране

Аналогичным образом следует поступать в среде Linux. После создания каталога и копирования в него файлов с компакт-диска программы можно запускать, вводя их имена в командной строке. При этом надо позаботиться, чтобы рабочий каталог присутствовал в пути запуска. Как и в Windows, при запуске программ в командной строке Linux можно задавать входные и выходные файлы.

Если имена программ или файлов с исходными данными некорректно воспроизводятся при копировании с компакт-диска в среде Linux, следует воспользоваться двумя tar-архивами, имеющимися на компакт-диске. Извлечь из них файлы можно командами

```
tar -xvf /cdrom/ARCe/Linux/apcedat.tar
tar -xvf /cdrom/ARCe/Linux/apceexe.tar
```

П.2. Компиляция и сборка программ

В этом разделе описываются шаги, необходимые для того, чтобы скомпилировать и собрать отдельные программы, а также приводятся некоторые советы по адаптации программ к требованиям различных операционных систем и компиляторов. Мы касаемся здесь только наиболее важных проблем, с которыми можно столкнуться в ходе компиляции. Остальную информацию обычно можно найти в документации по соответствующему компилятору.

П.2.1. Общие рекомендации по адаптации к компилятору

Хотя программы, написанные на C++, в общем случае могут выполняться на различных компьютерах, тем не менее между реализациями языка существуют небольшие различия. Одной из причин этого является введение в язык новых элементов (например, пространств имен, обработки исключений), а также расширение стандартных библиотек C++ в рамках работы по их стандартизации. В нашем случае чаще всего встречаются различия в реализации библиотеке <iostream>, что может потребовать небольших модификаций в программах.

Еще один источник различий — это аппаратно-программная архитектура используемых систем, и в первую очередь различия, связанные с диапазонами и точностью представления целых чисел и чисел с плавающей точкой. Мы предполагаем, что 32-битная операционная система поддерживает 4-байтовые целые числа (int) в диапазоне от –2 147 483 648 до +2 147 483 647 и 8-байтовые числа с плавающей точкой (double) с точностью 15–16 десятичных знаков. Тем не менее большинство функций и программ правильно работают, даже если компилятор использует 2-байтовые целые. Только в функции CalDat принципиально важно заменить в таком случае тип int на тип long.

Обычно не требуется вручную адаптировать программы к точности вычислений, характерной для вашей машины, так как программы сами получают необходимые сведения. Для этого служит предусмотренный в стандартной библиотеке C++ модуль <limits>, в котором есть функция numeric_limits<double>::epsilon. Однако в старых компиляторах C++ эту задачу может решать макрос DBL_EPSILON из <float.h> или функция

```
//GetAccuracy: Определяет машинную точность
double GetAccuracy() {
  double eps = 1.0e-6;
```

```
  while ( 1.0 + eps > 1.0 ) eps *= 0.5;
  return 2.0*eps;
}
```

которая определяет наименьшее число типа double, которое при сложении с 1,0 дает результат, отличный от 1,0. В этом случае стандартный тип данных double, имеющий 53-разрядную мантиссу и реализованный в соответствии с рекомендациями IEEE, выдаст в качестве относительной точности машинной арифметики $u \approx 2{,}2 \cdot 10^{-16}$.

П.2.2. Microsoft Visual C++ for Windows 95/98/NT

Опишем основные шаги, необходимые для получения исполняемых программ при использовании англоязычной версии среды разработки Microsoft Visual C++ 5.0.

Прежде всего создадим папку C:\APCe\Source\ (это можно сделать, например, при помощи Проводника Windows) и скопируем в нее все файлы из одноименной папки на прилагаемом к книге CD-ROM. После запуска среды разработки (MS Developer Studio) нужно создать (выбрав пункт меню **File ▸ New ▸ Workspaces**) новое рабочее пространство для управления отдельными программами и проектами и задать его название (рекомендуется Prj), а также рабочий каталог (рекомендуется C:\APCe\Prj).

Первым проектом в только что созданном рабочем пространстве будет библиотека модулей общего назначения, которые могут подключаться к различным приложениям. Это значительно упрощает разработку приложений. Выберите пункт меню **File ▸ New ▸ Projects** и создайте библиотеку типа *Win32 Static Library*, дав ей имя APC_Lib. Выберите вариант *Add to current workspace* и нажмите OK.

Теперь добавьте в проект APC_Lib библиотечные модули. Для этого выберите в меню **Project ▸ Add to Project ▸ Files** и отметьте в папке C:\APCe\Source все имена, соответствующе шаблону APC_*.cpp. При желании можно выделить и файлы *.h, но это не является принципиальным, так как компилятор сам найдет нужные файлы-заголовки. Теперь библиотеку можно создать командой **Build ▸ Build**. Среда разработки сначала скомпилирует все исходные файлы, а затем соберет их в статическую библиотеку APC_Lib.lib.

Все приложения — от AOEcat до Sunset можно теперь создавать по одной схеме в качестве независимых проектов в рамках рабочего пространства APCe\Prj. Выберем в качестве примера Coco. Командой **File ▸ New ▸ Projects** создаем проект типа *Win32 Console Application*. Теперь командой **Project ▸ Add to Project ▸ Files** добавляем в него файл Coco.cpp из каталога C:\APCe\Source и выбираем команду **Project ▸ Dependencies...** Для текущего проекта Coco выбираем APC_Lib, что указывает на необходимость связать Coco с этой библиотекой. Исполняемая программа Coco.exe будет создана по команде **Build ▸ Build** и может быть запущена командой **Build ▸ Execute**.

Более подробную информацию о работе с компилятором Microsoft C++ и средой Developer Studio можно найти в обширной документации, поставляемой с этими продуктами.

П.2.3. GNU C++ for Linux

Текущая версия компилятора GNU C++ 2.91, поставляемая в составе SuSE Linux версии 6.2, не поддерживает манипуляторы iostream в стандартной библиотеке C++. Это сказывается на всех программах, использующих операции right, fixed, showpos и noshowpos. В качестве «лекарства» в файле GNU_iomanip.h определены следующие функции:

```
#include <<iomanip>
#include <<iostream>
namespace {
ostream& left (ostream& o){o.setf(ios::left, ios::adjustfield); return o;};
ostream& right(ostream& o){o.setf(ios::right, ios::adjustfield); return o;};
ostream& fixed(ostream& o){o.setf(ios.: fixed, ios·:adjustfield); return o;};
ostream& showpos  (ostream& o){o.setf(ios··showpos); return o;}.
ostream& noshowpos(ostream& o){o.setf(ios:.noshowpos); return o;}.
}
```

Этот файл можно включить в исходный код командой

```
#ifdef __GNUC__  // Изменения для GNU C++
#include "GNU_iomanip.h"
#endif
```

Другая несовместимость касается определения машинной точности с использованием модуля <limits> из стандартной библиотеки, который не реализован в GNU C++. Это влияет на модули APC_DE.cpp и APC_Kepler.cpp, которые можно модифицировать следующим образом:

```
#ifdef __GNUC__   // Изменения для GNU C++
#include "float.h"
#else
#include <<limits>
#endif
...
#ifdef __GNUC__  // GNU C++ modifications
const double umach = DBL_EPSILON.
#else
const double umach = numeric_limits<<double>::epsilon().
#endif
```

Чтобы скомпилировать и собрать программы, рекомендуется прежде всего собрать все модули APC_* в библиотеку libAPC.a:

```
> g++ -c APC* cpp        # Компилировать все библиотечные модули
> g++ rc libAPC APC* o   # Создать библиотеку из объектных файлов
```

Теперь, если все исходные файлы лежат в одном каталоге, программы можно скомпилировать и собрать следующими командами:

```
> g++ AOEcat.cpp -o AOEcat -lAPC -L.  # Компилируем и собираем AOEcat
> g++ Coco.cpp -o AOEcat -lAPC -L.    # Компилируем и собираем Coco
> ...
> g++ Sunset.cpp -o AOEcat -lAPC -L   # Компилируем и собираем Sunset
```

Для упрощения этого процесса в архиве apcesrc.tar на компакт-диске имеется make-файл, выполняющий все описанные шаги при вводе команды make.

В случае если имена упомянутых файлов некорректно прочитались с компакт-диска под Linux или другой UNIX-подобной системой (например, если драйвер компакт-диска не поддерживает расширенного стандарта ISO 9660), можно извлечь все необходимые исходные тексты из tar-архива командой

```
tar -xvf /cdrom/APCe/Linux/apcsrc.tar
```

П.3. Список библиотечных функций

Библиотека APC_Lib содержит базовый набор операций, функций и классов, которые не связаны с конкретными программами и могут поэтому использоваться отдельно для разработки других программ. Она состоит из 14 модулей, каждый из которых решает определенную группу задач:

APC_Cheb.h	Аппроксимация функции полиномами Чебышева
APC_Const.h	Математические и астрономические константы
APC_DE.h	Численное интегрирование дифференциальных уравнений
APC_IO.h	Вспомогательные функции для ввода и вывода
APC_Kepler.h	Кеплеровские орбиты
APC_Math.h	Технические и научные функции
APC_Moon h	Координаты Луны
APC_Phys h	Физические эфемериды
APC_Planets.h	Координаты планет
APC_PrecNut.h	Прецессия и нутация
APC_Spheric.h	Сферическая астрономия
APC_Sun.h	Координаты Солнца
APC_Time.h	Операции со временем и календарем
APC_VecMat3D.h	Векторные и матричные операции

Использование файлов-заголовков означает, что соответствующие модули будут подключены к создаваемому приложению. Реализации объявляемых в заголовках функций содержатся в соответствующих файлах APC_*.cpp, которые можно отдельно скомпилировать и подключать в виде объектных модулей или скомпоновать в библиотеку. Единственное исключение — модуль APC_Const, который не требует реализации в виде cpp-файла, а целиком состоит из файла-заголовка.

Далее приведена сводка имен типов, констант, функций и классов, которые определены в библиотечных модулях и доступны для использования

<<	операция вывода для дат, времен, углов, векторов и матриц
[]	доступ к компонентам вектора (декартовым, полярным)
+	сложение векторов и матриц
+=	добавление вектора
-	вычитание векторов и матриц
-	унарный минус (векторы, матрицы)
-=	вычитание вектора

*	умножение (скаляры, векторы, матрицы)
/	деление на скаляр
Accumulate	метод класса SolverLSQ для добавления данных уравнения
AddThe	теорема сложения для тригонометрических функций
Angle	вспомогательный класс для вывода углов
AngleFormat	тип-перечисление для задания формата вывода угла
APC_MaxFilename	максимальная длина имени файла в командной строке
Arcs	количество угловых секунд в радиане
AU	астрономическая единица [км]
Bright	видимая звездная величина планеты
c_light	скорость света [а. е./сут]
CalDat	календарная дата и время
Cheb3D	чебышевская аппроксимация в трех измерениях
Col	векторы-столбцы в матрице
Cross	векторное произведение
DateTime	вспомогательный класс для вывода даты и времени
Dd	значение типа AngleFormat — десятичное представление
DDd	значение типа TimeFormat — представление в долях суток
Ddd	доли градуса по минутам и секундам
DE_ACCERACY_NOT_ACHIEVED	значение типа DE_STATE (затребована слишком высокая точность)
DE_DONE	значение типа DE_STATE (шаг интегрирования успешно выполнен)
DE_INIT	значение типа DE_STATE (перезапуск интегратора)
DE_INVALID_PARAMS	значение типа DE_STATE (некорректный ввод)
DE_STATE	тип-перечисление для представления состояния интегратора SolverDE
DE_STIFF	значение типа DE_STATE (вероятно, жесткое дифференциальное уравнение)
DE_TO_MANY_STEPS	значение типа DE_STATE (слишком большое количество шагов)
Deg	$180°/\pi$
Direct	значение типа RotationType (прямое вращение)
DMM	значение типа AngleFormat для вывода градусов и целого числа минут
DMMm	значение типа AngleFormat для вывода градусов и минут с десятичными долями
DMMSS	значение типа AngleFormat для вывода градусов, минут и целого числа секунд
DMMSSs	значение типа AngleFormat для вывода градусов, минут и секунд с десятичными долями
DMS	минуты и секунды по долям градуса
Dot	скалярное произведение двух векторов
Earth	значение типа PlanetType (Земля)
EccAnom	эксцентрическая аномалия эллиптической орбиты
Ecl2EquMatrix	матрица преобразования эклиптических координат в экваториальные
Elements	элементы орбиты по положению и скорости
Elements	элементы орбиты по двум векторам положения

Ellip	положение и скорость на эллиптической орбите
Equ2EclMatrix	матрица преобразования экваториальных координат в эклиптические
Equ2Hor	преобразование экваториальных координат в горизонтальные
EquStd	стандартные координаты из экваториальных координат
ETminUT	аппроксимация разности ET–UT
FileExists	проверка существования файла
FindEta	отношение сектор–треугольник
Fit	метод класса Cheb3D для чебышевской аппроксимации функции
Frac	дробная часть числа
GaussVec	матрица преобразования координат от плоскости орбиты к эклиптике
GetFilenames	выделение имени файла из командной строки
GM_Sun	произведение гравитационной постоянной и массы Солнца [а. е./сут]
GMST	гринвичское среднее звездное время
HHh	значение типа TimeFormat — вывод времени в десятичных долях часа
HHMM	значение типа TimeFormat — вывод времени в часах и минутах
HHMMSS	значение типа TimeFormat — вывод времени в часах, минутах и секундах
Hor2Equ	преобразование горизонтальных координат в экваториальные
HypAnom	эксцентрическая аномалия для гиперболических орбит
Hyperb	положение и скорость на гиперболической орбите
Id3D	единичная матрица
Illum	параметры освещенности планет
index	тип-перечисление для доступа к компонентам объектов типа Vec3D
Init	метод класса SolverLSQ для инициализации задачи аппроксимации
Integ	метод класса SolverDE для выполнения шага интегрирования
Jupiter	значение типа PlanetType (Юпитер)
JupiterPos	гелиоцентрические координаты Юпитера
Kepler	положение и скорость на кеплеровой орбите
KepPosition	положение планеты по элементам кеплеровой орбиты
KepVelocity	скорость планеты по элементам кеплеровой орбиты
kGauss	гауссова гравитационная постоянная [а. е.^{3}/сут^2]
Mars	значение типа PlanetType (Марс)
MarsPos	гелиоцентрические координаты Марса
Mat3D	класс матриц 3×3
Mercury	значение типа PlanetType (Меркурий)
MercuryPos	гелиоцентрические координаты Меркурия
MiniMoon	координаты Луны с низкой точностью
MiniSun	координаты Солнца с низкой точностью
Mjd	модифицированная юлианская дата
MJD_J2000	модифицированная юлианская дата для эпохи 2000,0

Modulo	вычисление остатка от деления
MoonEqu	экваториальные координаты Луны с высокой точностью
MoonPos	эклиптические координаты Луны с высокой точностью
Neptune	значение типа PlanetType (Нептун)
NeptunePos	гелиоцентрические координаты Нептуна
None	значение типа TimeFormat — вывод даты без времени
Norm	длина вектора
NutMatrix	матрица нутации
Orient	ориентация планетоцентрической системы координат
Parab	положение и скорость для параболической орбиты
Pegasus	поиск корней с использованием метода Пегаса
PertPosition	положения планет с высокой точностью
phi	значение типа pol_index для получения азимута вектора
pi	π
pi2	2π
PlanetType	тип-перечисление — Солнце и планеты
Pluto	значение типа PlanetType (Плутон)
PlutoPos	гелиоцентрические координаты Плутона
pol_index	тип-перечисление для доступа к полярным компонентам вектора
Polar	вспомогательный класс для задания вектора полярными координатами
PosAng	позиционный угол
PrecMatrix_Ecl	матрица прецессии (эклиптические координаты)
PrecMatrix_Equ	матрица прецессии (экваториальные координаты)
Quad	квадратичная интерполяция
r	значение типа pol_index для получения длины вектора
R_Earth	радиус Земли [км]
R_Moon	радиус Луны [км]
R_Sun	радиус Солнца [км]
R_x	матрица элементарного поворота вокруг оси x
R_y	матрица элементарного поворота вокруг оси y
R_z	матрица элементарного поворота вокруг оси z
Rad	$\pi/180°$
Retrograde	значение типа RotationType (обратное вращение)
Rotation	планетографические координаты
RotationType	тип-перечисление для задания направления вращения
Row	вектор-строка в матрице
Saturn	значение типа PlanetType (Сатурн)
SaturnPos	гелиоцентрические координаты Сатурн
Set	метод классов Angle, Time и DateTime для выбора формата вывода
Shape	формы и размеры планет
Site	координаты точки на поверхности Земли
Solve	метод обратной подстановки (член класса SolverLSQ)
SolverDE	класс для численного решения дифференциальных уравнений
SolverLSQ	класс для решения задачи аппроксимации

StdEdu	вычисление экваториальных координат по стандартным координатам
Stumpff	функции Штумпфа
Sun	значение типа PlanetType (Солнце)
SunEqu	экваториальные координаты Солнца
SunPos	эклиптические координаты Солнца
Sys_I	значение типа SystemType (система I)
Sys_II	значение типа SystemType (система II)
Sys_III	значение типа SystemType (система III)
SystemType	тип-перечисление для задания вращающейся системы отсчета
T_B1950	эпоха B1950 (в юлианских столетиях от J2000)
T_J2000	эпоха J 2000 (в юлианских столетиях от J2000)
theta	значение типа pol_index для высоты вектора
Time	вспомогательный класс для вывода времени
TimeFormat	тип-перечисление для задания формата вывода времени и даты
Transp	транспонированная матрица
Uranus	значение типа PlanetType (Уран)
UranusPos	гелиоцентрические координаты Уран
Value	метод класса Cheb3D для получения значения с использованием чебышевской аппроксимации
Vec3D	класс трехмерных векторов
Venus	значение типа PlanetType (Венера)
VenusPos	гелиоцентрические координаты Венеры
x	значение типа index для получения x-координаты вектора
y	значение типа index для получения y-координаты вектора
z	значение типа index для получения z-координаты вектора

Список обозначений

a	большая полуось
a	дифференциальный коэффициент по долготе
A	азимут
$\boldsymbol{A}$	матрица коэффициентов системы линейных уравнений
b	эклиптическая долгота
b	дифференциальный коэффициент по широте
$\dot{b}$	изменение эклиптической широты со временем
$\boldsymbol{b}$	вектор системы линейных уравнений
B	эклиптическая широта Солнца
c	скорость света (c = 299 792 458 м/с)
$c_i(E)$	функция Штумпфа
C	параллактический угол
$\boldsymbol{d}$	диаметр земной тени
$\boldsymbol{d}$	вектор системы линейных уравнений в тригонометрической форме
d	единичный вектор, совпадающий с осью вращения
db	возмущение по эклиптической широте
dl	возмущение по эклиптической долготе
dr	возмущение по расстоянию
D	средняя элонгация Луны
D	диаметр лунной полутени на поверхности Земли
e	эксцентриситет
$\boldsymbol{e}$	единичный вектор
E	эксцентрическая аномалия
E	элонгация
$\boldsymbol{E}$	матрица преобразования
ET	эфемеридное время
f	сжатие Земли
f	половина угла при вершине конуса тени и полутени
f	разность координат в главной плоскости
F	сила притяжения между двумя телами
F	среднее расстояние Луны от восходящего узла ее орбиты
F	фокусное расстояние
g	разность координат в главной плоскости
G	гравитационная постоянная ($G = 2{,}959122083 \cdot 10^{-4}$ а. е.$^3 M_\odot^{-1}$d^{-2})
GMST	среднее звездное время по Гринвичу (Greenwich Mean Sidereal Time)

h	высота
h	величина шага
H	эксцентрическая аномалия для гиперболической орбиты
i	наклонение орбиты к эклиптике
i	угол фазы
$\boldsymbol{i}$	единичный вектор в главной плоскости (ось x)
$\boldsymbol{j}$	единичный вектор в главной плоскости (ось y)
JD	юлианская дата
k	Гауссова гравитационная постоянная (k = 0,01720209895)
k	фаза
k	отношение радиуса Луны к радиусу Земли ($k \approx 0,2725$)
$\boldsymbol{k}$	единичный вектор, перпендикулярный главной плоскости (ось z)
l	эклиптическая долгота
l	радиус тени в главной плоскости
l	средняя аномалия Луны
l'	средняя аномалия Солнца
$\dot{l}$	изменение эклиптической долготы со временем
L	эклиптическая долгота Солнца
L	радиус тени в пункте наблюдения
L_0	средняя долгота Луны
m	видимая звездная величина
m	расстояние между наблюдателем и осью лунной тени
M	средняя аномалия
M	наибольшая фаза солнечного затмения
M_{h}	средняя аномалия (гиперболическая орбита)
$M_{\odot}$	масса Солнца
MJD	модифицированная юлианская дата
n	отношение площадей треугольников
n	суточное движение
$\boldsymbol{N}$	матрица нутации
p	фокальный параметр
P	позиционный угол точки контакта солнечного затмения
$\boldsymbol{P}$	Гауссов вектор (направление на перигелий)
$\boldsymbol{P}$	матрица прецессии
q	расстояние перигелия
$\boldsymbol{Q}$	Гауссов вектор (перпендикулярный направлению на перигелий)
r	расстояние
$\boldsymbol{r}$	радиус-вектор
$\dot{r}$	изменение расстояния
R	расстояние от Земли до Солнца
R	радиус небесного тела
R	редукция к эклиптике

R	рефракция
$\boldsymbol{R}$	Гауссов вектор, перпендикулярный к плоскости орбиты
$\boldsymbol{R}$	вектор геоцентрических координат Солнца
$\boldsymbol{R}$	верхнетреугольная матрица
$\boldsymbol{R}_{x,y,x}$	матрица элементарных поворотов
S	площадь сектора
t	время
T	время в юлианских столетиях, прошедшее после эпохи J2000
T	период обращения
T_0	время прохождения перигелия
$T_n(x)$	полином Чебышева n-го порядка
ΔT	разница между эфемеридным и Всемирным временем
TDB	барицентрическое динамическое время
TDT	земное динамическое время
u	аргумент широты
U	вспомогательная переменная для орбит, близким к параболическим
$\boldsymbol{U}$	матрица преобразования
UT	Всемирное время
UTC	Всемирное координатное время
$\boldsymbol{v}$	вектор скорости
V	позиционный угол точки контакта солнечного затмения
$V(1,0)$	видимая звездная величина на единичном расстоянии
W	ориентация начального меридиана
x, y, z	Декартовы координаты
$\boldsymbol{y}$	вектор состояния
X, Y	стандартные координаты в астрометрии
z	вспомогательный угол для описания прецессии
α	прямое восхождение
α_0	прямое восхождение оси вращения
β	эклиптическая широта
γ	косинус элонгации
δ	склонение
δ_0	склонение оси вращения
Δ	геоцентрическое расстояние
Δ	площадь треугольника
ε	наклон эклиптики
$\Delta\varepsilon$	нутация по эклиптической широте
ζ	вспомогательный угол для описания прецессии
η	отношение площадей сектора и треугольника
ϑ	сферическая координата
ϑ	вспомогательный угол для описания прецессии

ϑ	позиционный угол
Θ	местное звездное время
Θ_0	звездное время по Гринвичу
λ	эклиптическая долгота
λ	географическая долгота (положительное направление на восток)
λ	планетографическая долгота (положительное направление, противоположное вращению)
λ'	планетоцентрическая долгота (положительное направление к востоку)
Λ	вспомогательный угол для описания прецессии
μ	вспомогательная величина для выражения соотношения площадей треугольников
ν	истинная аномалия
π	3,1415926...
π	экваториальный горизонтальный параллакс
π	вспомогательный угол для описания прецессии
Π	вспомогательный угол для описания прецессии
ϖ	долгота перигелия
ρ	геоцентрическое расстояние
σ	параметр в уравнении Гаусса—Лагранжа
τ	часовой угол
τ	интервал
φ	сферическая координата
φ	географическая (планетографическая) широта
φ'	геоцентрическая (планетоцентрическая) широта
ξ	остаточная сумма площадей
$\Delta\psi$	нутация по эклиптической широте
ω	аргумент перигелия
☊	долгота восходящего узла
∅	диаметр
♈	точка весеннего равноденствия
⊕	Земля
⊙	Солнце
^{d}	сутки
^{h}	часы
^{m}	минуты времени
^{m}	звездная величина
^{r}	обороты
^{s}	секунды времени
$^\circ$	градусы дуги

Словарь терминов

Аберрация — смещение видимого положения небесного тела относительно его истинного положения, обусловленное конечностью скорости света (*аберрация света, время распространения света*).

Аберрация света — разность между видимым направлением на светило для неподвижного относительно Солнца наблюдателя и для наблюдателя, движущегося вместе с Землей.

Азимут (астрономический) — одна из координат в горизонтальной системе координат. Азимут — это угол, измеряемый вдоль горизонта и отсчитываемый от точки юга (в отличие от геодезического азимута) в сторону запада, — то есть угол между направлением, в котором наблюдается объект, и направлением на юг.

Астрометрические координаты — координаты светила в форме, пригодной для сравнения с каталожными положениями звезд. Астрометрические координаты обычно относятся к среднему *весеннему равноденствию* стандартной эпохи (*J2000, B1950*), в случае планет или комет учитывается также *время распространения света.*

Астрометрия — измерение координат звезд различными методами, традиционными являются измерения с помощью инструментов, снабженных разделенными кругами (имеющими градуировку), и измерения фотопластинок.

Астрономическая единица — единица длины, используемая для выражения расстояний в Солнечной системе. Одна астрономическая единица (а. е.) соответствует среднему расстоянию от Земли до Солнца и составляет примерно 149,6 млн км.

Видимые положения — координаты небесного тела, знание которых требуется, например, для наведения на объект телескопа, снабженного разделенными кругами. Видимые положения связаны с действительным направлением земной оси, поэтому требуется учитывать *прецессию* и *нутацию*. Также должны учитываться *аберрация света*, а в случае тел Солнечной системы и *время распространения света.*

Восходящий узел — точка орбиты, в которой небесное тело пересекает эклиптику с юга на север.

Время распространения света — время, за которое свет, испущенный светилом, преодолевает расстояние до Земли (499 с на астрономическую единицу). За это время планета несколько смещается по орбите, и ее видимое положение не соответствует истинному положению в момент наблюдения.

Всемирное время — система счета времени, связанная с вращением Земли, на которой основано гражданское время. Всемирное время может быть определено из

наблюдений прохождения через меридиан небесных тел с *известным прямым восхождением*. Из-за неравномерного вращения Земли Всемирное время подвержено нерегулярным вариациям и, в отличие от эфемеридного и динамического времени, не является равномерным.

Высота — угол между направлением на небесное тело и *горизонтом*. Видимая высота объекта из-за *рефракции* может отличаться от действительной на величину, достигающую 35′.

Гелиографические координаты — система координат для однозначного описания точки на видимой поверхности Солнца, определяемая аналогично *географическим* и *планетографическим* координатам.

Гелиоцентрические координаты — система координат, связанная с центром Солнца.

Географические координаты — два числа (географические широта и долгота), указывающие положение точки на поверхности Земли, в системе координат, связанной с плоскостью земного экватора. Долгота отсчитывается на восток от нулевого — Гринвичского *меридиана*, определенного международным соглашением.

Геоцентрические координаты — система координат, связанная с центром Земли.

Горизонт — воображаемая линия пересечения плоскости, касательной к поверхности Земли в точке, где находится наблюдатель, и небесной сферы. С плоскостью горизонта связана *горизонтальная система координат* (*азимут* и *высота*).

Горизонтальная система координат — система координат, связанная с местным горизонтом в пункте наблюдения, в которой положение точки характеризуется *азимутом* и *высотой*. Горизонтальная система координат используется, например, при вычислении восхода и захода небесных тел. Однако она не пригодна для эфемерид, поскольку азимут и высота светила зависят от места наблюдения и непрерывно изменяются вследствие вращения Земли.

Динамическое время (TDB, TDT) — система счета времени, основанная на периодических процессах, не связанных с вращением Земли, как, например, в атомных часах. Динамическое время, как и *эфемеридное время*, однородно, тем не менее требуется учитывать релятивистские поправки. Если барицентрическое динамическое время (TDB) соответствует показаниям часов в центре тяжести Солнечной системы, то земное динамическое время (TDT) соответствует показаниям часов в центре Земли. Разница между этими двумя системами счета времени составляет всего несколько миллисекунд, такого же порядка их отличие от *эфемеридного времени*. В 1991 году было принято новое название земного динамического времени (TDT) — земное время (TT).

Задача двух тел — задача вычисления движения двух тел под действием силы взаимного притяжения. Ее решение дает упрощенное представление движения планеты, кометы или астероида вокруг Солнца, если пренебречь возмущениями со стороны других тел Солнечной системы. Относительная простота ее математического решения имеет большое значение для вычисления эфемерид.

Задача n-тел — задача вычисления движения более двух тел под действием сил взаимного притяжения. Если задача двух тел имеет аналитическое решение, задача n-тел такого решения не имеет в принципе. Для этого используются численные методы и аналитические приближения (разложение в ряд).

Звездное время — прямое восхождение звезды в верхней *кульминации*, то есть находящейся в точности в меридиане. Звездные сутки соответствуют одному обороту земли и составляют примерно 23^h56^m Всемирного времени.

Земное время (TT) — релятивистская система счета времени, в настоящее время вытесняет *эфемеридное время* (ET) как основу геоцентрических эфемерид (см. *динамическое время*). Для практических целей обе системы счета времени могут считаться эквивалентными.

Зенит — воображаемая точка на небесной сфере точно над наблюдателем (точка пересечения отвесной линии и небесной сферы). Высота точки зенита, таким образом, равна 90°.

Кеплера задача — то же, что и задача двух тел.

Кульминация — момент прохождения светилом небесного *меридиана*, когда *высота* светила над горизонтом достигает наибольшего или наименьшего значения.

Меридиан — (в астрономии) большой круг на небесной сфере, проходящий через *зенит* и пересекающий *горизонт* в точках севера и юга (см. также *кульминация*). Географический меридиан — половина большого круга на поверхности Земли, лежащего в плоскости ее оси. Все меридианы проходят через полюсы и пересекают экватор под прямым углом. Меридианы являются линиями постоянной долготы.

Нутация — периодические колебания земной оси относительно ее среднего положения, налагающиеся на *прецессию*. Период нутации составляет 18,6 лет и обусловлен периодом обращения линии узлов лунной орбиты.

Главный меридиан — избранный меридиан, от которого отсчитывается долгота в планетографической системе координат.

Параллакс — смещение видимого положения небесного тела относительно неподвижных звезд поля, вызванное изменением положения наблюдателя. Параллакс имеет наибольшее значение для близких объектов и заметнее всего у Луны и в гораздо меньшей степени у комет и астероидов, проходящих вблизи Земли (см. также *топоцентрические координаты*).

Перигей — ближайшая к Земле точка орбиты Луны или спутника.

Перигелий — ближайшая к Солнцу точка орбиты планеты или кометы.

Перифокальные координаты — положение небесного тела относительно плоскости его орбиты и лини апсид.

Планетографические координаты — угловые величины, описывающие положение точки на поверхности планеты аналогично *географическим координатам* на Земле.

Позиционный угол — угол на небесной сфере между некоторым избранным направлением и направлением на заданную точку (например, полюс планеты). Обычно позиционный угол отсчитывается от направления на север к востоку и измеряется в градусах.

Прецессия — долгопериодическое изменение положения эклиптики и небесного экватора (*точка весеннего равноденствия* смещается вдоль небесного экватора в сторону уменьшения долгот). Вследствие возмущающего влияния Солнца и Луны, земная ось с периодом в 26 000 лет описывает в пространстве конус, ось которого направлена на средний полюс эклиптики. Поэтому положение небесного экватора непрерывно изменяется. Возмущения от других планет вызывают также медленное изменение положения эклиптики. Движение экватора и эклиптики, смещение точки весеннего равноденствия означает, что *экваториальные* и *эклиптические координаты* объекта должны быть сопровождены указанием даты, к которой они относятся. Обычно координаты относятся к равноденствию даты, эпохе *B1950* или *J2000.*

Прямое восхождение — в *экваториальной системе координат* угол, отсчитываемый вдоль небесного *экватора* к востоку от *точки весеннего равноденствия.* Величина угла обычно выражается в часовой мере ($15° \equiv 1^h$) и измеряется в часах, минутах и секундах.

Равноденствие — см. *Точка весеннего равноденствия*[1].

Рефракция — преломление лучей света в земной атмосфере. Для наблюдателя на поверхности Земли небесное тело имеет несколько большую высоту над горизонтом, нежели при отсутствии рефракции. Вблизи горизонта лучи света проходят значительную толщу атмосферы, поэтому рефракция наиболее заметна при восходе или заходе светила.

Склонение — угол на небесной сфере между направлением на светило и небесным экватором, достигающий 90°. Вместе с прямым восхождением образует *экваториальную систему координат.* (См. также *точка весеннего равноденствия.*)

Топоцентрические координаты — координаты, связанные с положением наблюдателя на поверхности Земли. Топоцентрические и геоцентрические координаты различаются на величину параллакса.

Точка весеннего равноденствия — точка пересечения *эклиптики* и *небесного экватора.* Солнце, перемещаясь с юга на север в своем годичном движении по эклиптике, пересекает экватор в этой точке, что происходит ежегодно примерно 21 марта. Этот момент определяет наступление весны. Точка весеннего равноденствия служит отправной точкой при определении эклиптической долготы (см. *эклиптические координаты*) и *прямого восхождения.* Поскольку положение эклиптики и небесного экватора меняется с течением времени из-за *прецессии,* координаты должны сопровождаться указанием даты, к которой они относятся. Обычно используются равноденствие даты, эпохи *B1950* и *J2000.*

---

[1] В англоязычной литературе термин «равноденствие» часто используется также как синоним термина «эпоха». — *Примеч. ред.*

Эклиптика — большой круг, образованный пересечением плоскости земной орбиты и небесной сферы. При наблюдении с Земли Солнце совершает годичный путь вдоль эклиптики. С ней связана эклиптическая система координат, в которой положение точки определяется эклиптическими широтой и долготой (см. *эклиптические координаты*).

Эклиптические координаты — координаты небесного тела, отнесенные к *эклиптике* и равноденствию некоторой эпохи, представленные в виде широты и долготы. Эклиптическая долгота отсчитывается к востоку от *точки весеннего равноденствия*. Эклиптическая широта представляет собой угол, отсчитываемый перпендикулярно эклиптике, между ней и направлением на светило.

Элонгация — видимое угловое расстояние между двумя объектами.

Эфемериды — таблицы, содержащие координаты планеты, кометы или астероида для определенного периода времени.

Эфемеридное время (ЕТ) — система счета равномерного времени, используемая для вычисления координат планеты, кометы или астероида. Эфемеридное время свободно от нерегулярностей и непредсказуемых неравномерностей вращения Земли, на котором основано измерение *Всемирного времени*. Разница между эфемеридным (ЕТ) и Всемирным (UT) временем была принята равной нулю в начале XX века и в настоящее время составляет около одной минуты. В рамках релятивистской теории эфемеридное время заменено *земным временем (TT)*.

Экватор (небесный) — большой круг на небесной сфере, перпендикулярный земной оси. Экватор разделяет северную и южную полусферы, и с его плоскостью связана экваториальная система координат, в которой направление на светило представлено *прямым восхождением* и *склонением*.

Литература

Мы приводим список работ, посвященных сферической астрономии, небесной механике и связанным с ними аспектам математики и программирования. С их помощью читатель может глубже изучить отдельные предметы, детальное рассмотрение которых было невозможно в настоящей книге. По возможности мы приводим дополнительные ссылки на www-сайты. Однако в силу своей природы адреса в Интернете имеют тенденцию часто изменяться. Поэтому, если приведенные адреса устареют, мы рекомендуем читателю использовать доступные средства поиска (AltaVista, Lycos, Yahoo! и др.).

Общие работы

[1] *Astronomical Almanac*; U.S. Government Printing Office, Her Majesty's Stationery Office; Washington, London.

Ведущий англоязычный календарь. Выпуск 1984 года содержит рекомендованные MAC для астрономических вычислений постоянные и численные значения различных величин (звездное время, прецессия и нутация, массы и диаметры планет, элементы для физических эфемерид).

[2] H. Bucerius, M. Schneider; *Himmelsmechanik I-II*; Bd. 143/144, Bibliographishes Institut; Mannheim (1966).

Современное изложение небесной механики. Среди прочего рассматриваются общая теория планетных возмущений, теория движения Луны и вычисление орбит.

[3] L. E. Doggett, G. H. Kaplan, P. K. Seidelmann; *Almanac for Computers for the Year 19xx*; Nautical Almanac Office, United States Naval Observatory; Washington.

Ежегодное издание, содержащее описание положений Солнца, Луны и планет в виде полиномов Чебышева, позволяющее легко вычислять любые эфемериды с достаточной точностью даже с помощью небольших компьютеров. Вводная глава содержит рекомендации и простые формулы для вычисления астрономических явлений.

[4] P. K. Seidelmann (ed.); *Explanatory Supplement to the Astronomical Almanac*; University Science Books (1992).

Приложение к ежегоднику *Astronomical Almanac*, содержащее подробное описание фундаментальных данных и использованных методов вычислений. Оно заменяет *Explanatory Supplement to the Astronomical Ephemeris*

and the American Ephemeris and Nautical Almanac, опубликованный U. S. Government Printing Office, Washington и Her Majesty's Stationery Office, London, 1974, впрочем, в нем содержатся не все материалы из последнего издания.

[5] R. M. Green; *Spherical Astronomy*; Cambridge University Press; Cambridge (1985).

Современное изложение сферической астрономии.

[6] A. Guthmann; *Einführung in die Himmelsmechanik und Ephemeridenrechung*; Spectrum Akademie Verlag (ehem. BI Wissenschaftsverlag), Heidelberg (1994).

Введение и практический курс небесной механики и вычисления эфемерид.

[7] D. McNally; *Positional Astronomy*; Muller Educational; London (1974).

Введение в сферическую астрономию.

[8] J. Meeus; *Astronomical Algorithms*; Willmann-Bell; Richmond, Virginia (1991).

Исправленное и существенно дополненное издание *Astronomical Formulae for Calculators*. Предназначенное для практических целей собрание любопытных и необычных вычислительных процедур для любителей астрономии. Содержит разнообразные примеры, доступные для решения на небольших компьютерах. Доступны также гибкие диски с программами на Бейсике, Паскале или C.

[9] J. Meeus; *Astronomical Tables of the Sun, Moon and Planets*; Willmann-Bell; Richmond, Virginia (1983).

Сборник разнообразных любопытных астрономических сведений, включая фазы Луны, солнечные и лунные затмения, соединения планет, покрытия планет Луной и т. п.

[10] I. I. Mueller; *Spherical and practical astronomy*; Frederick Ungar Publishing Co.; Ney York (1969).

Изложение сферической астрономии и ее приложений в геодезии. Рассматриваются также способы определения времени и их применение на практике, предвычисление солнечных затмений и покрытий звезд Луной.

[11] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery; *Numerical Recipes in C*; Cambridge University Press; Cambridge, 2nd ed. (1993).

Полное собрание важных процедур, охватывающих все области математических вычислений. Существуют также издания для Фортрана и Паскаля. Исходные коды доступны на разнообразных носителях.

[12] A. E. Roy; *Orbital Motion*; Adam Hilger Ltd.; Bristol, 2nd ed. (1982).

Общее введение в небесную механику.

[13] H. R. Schwarz; *Numerishe Matematik*; B. G. Teubner Verlag; Stuttgart, 2. Aufl. (1988).

Изложение важных алгоритмов математических вычислений, предназначенное для решения практических задач.

[14] K. Stumpff; *Himmelsmechanik I-III*; VEB Deutsher Verlag der Wissenschaften; Berlin (1959, 1965, 1974).

Изложение небесной механики.

[15] B. Stroustrup; *The C++ Programming Language*; Addison Wesley; 3rd edition (1997).

Полный курс, составленный разработчиком C++. Отдельно рассматриваются современный языковой стандарт и библиотека стандартных шаблонов.

Глава 2

[16] J. H. Lieske, T. Lederle, W. Fricke, B. Morando; *Expressions for the Precession Quantities Based upon the IAU (1976) System of Astronomical Constants*; Astronomy and Astrophysics, vol. 58, pp. 1–16 (1977).

Приводятся различные вспомогательные величины для описания прецессии в эклиптических и экваториальных координатах. Численные значения основаны на системе астрономических постоянных МАС, 1976.

[17] G. Moyer; *The Origin of the Julian Day System*; Sky and Telescope, vol. 61, pp. 311–313 (April 1982).

Статья по истории юлианской даты, содержащая формулы для перехода от номера текущей даты к обычному календарю.

Глава 3

[18] L. D. Schmadel, G. Zech; *Empirical transformations from UT to ET for the period 1800–1988*; Astronomishe Nachrichten, vol. 309, pp. 219–221 (1988).

Наблюдаемая разница между Всемирным и эфемеридным временем представлена в виде полинома 12-го порядка, пригодного для вычислений в пределах промежутка, покрывающего почти двести лет.

Алгоритмы для вычислений времени восхода и захода светил и для учета рефракции приведены в *Almanac for Computers* [3].

Глава 4

[19] J. M. A. Danby, T. M. Burkhardt; *The solution of Kepler's Equation I–III*; Celestial Mechanics, vol. 31, pp. 95–107, pp. 317–328 (1983).

Описание различных итерационных процедур для решения уравнения Кеплера, приводятся соответствующие начальные значения.

[20] B. Marsden; *Catalogue of Cometary Orbits*; Smithsonian Astrophysical Observatory; Cambridge, Mass.

Периодически обновляемый каталог оскулирующих элементов орбит всех известных комет.

[21] Ephemerides of Minor Planets; Institute for Theoretical Astronomy; St. Petersburg.

Элементы орбит и эфемериды известных малых планет. Публикуется ежегодно.

[22] Yu. V. Batrakov, V. A. Shor; Catalogue of Orbital elements and Photometric parameters of 7316 Minor Planets numbered by 25 November, 1996; Institute for Theoretical Astronomy; St. Petersburg, Russia (1997).

Элементы орбит на текущий момент для наиболее значимых каталогизированных малых планет. Доступны по адресу Центра астрономических данных НАСА:

http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1245/

Методы Штумпфа для вычисления почти параболических орбит описаны в *Himmelsmechanik I* [14]. Исчерпывающая база данных по элементам орбит малых планет Лоуэлловской обсерватории находится по адресу ftp:// ftp.lowell.edu/pub/elgb/astorb.html.

Глава 5

[23] L. F. Shampine, M. K. Gordon; *Computer Solution of Ordinary Differential Equations*; Freeman and Comp., San Fransisco (1975).

Введение в теорию многоступенчатых методов численного решения обычных дифференциальных уравнений и описание метода Адамса с переменным шагом.

[24] X. X. Newhall, E. M. Standish Jr., J. G. Williams; *DE102: a numerically integrated ephemeris of the moon and planets spanning forty-four centuries*; Astronomy and Astrophysics, vol. 125, pp. 150–167 (1983).

Лучшие эфемериды для периода в несколько тысяч лет. Приведенные алгоритмы пригодны и для более поздних версий JPL Development Ephemerides. Современные версии (DE200, DE405 и т. д.) доступны по адресу http://ssd.jpl.nasa.gov/eph_info.html.

Обновленные элементы орбит малых планет ежегодно публикуются в *Ephemerides of Minor Planets* ([21], см. также [22]) Института теоретической астрономии в Санкт-Петербурге, а также в различных астрономических календарях.

Глава 6

[25] P. Bretagnon, J.-L. Simon; *Tables for the motion of the sun and the five bright planets from −4000 to +2800; Tables for the motion of Uranus and Neptune from +1600 to +2800*; Willmann-Bell; Richmond, Virginia (1986).

Разложение в ряд и программы, с помощью которых движение внутренних планет может быть вычислено с достаточной точностью. Координаты внешних планет представлены полиномами.

[26] T. C. Van Flandern, K. F. Pulkkinen; *Low precision formulae for planetary positions*; Astrophysical Journal Supplement Series, vol. 41, p. 391 (1979).

Разложение в ряд для грубых (~1′) вычислений гелиоцентрических и геоцентрических координат планет.

[27] E. Goffin, J. Meeus, C. Steyaert; *An accurate representation of the motion of Pluto*; Astronomy and Astrophysics, vol. 155, p. 323 (1986).

Представление орбиты Плутона тригонометрическими рядами, полученными из численно проинтегрированных эфемерид.

[28] G. W. Hill; *Tables of Jupiter, Tables of Saturn*; Astronomical Papers of the American Ephemeris, vol. VII, part 1–2; Washington (1898).

Аналитическое разложение в ряд движения Юпитера и Сатурна.

[29] M. P. Jarnagin, jr.; *Expansions in elliptic motion*; Astronomical Papers of the American Ephemeris, vol. XVIII; Washington (1965).

Разложение в ряд для уравнения центра и радиуса, а также других координат в невозмущенной задаче Кеплера при высоких порядках эксцентричности орбиты.

[30] S. Newcomb; *Tables of the motion of the Earth, Tables of the heliocentric motion of Mercury, Tables of the heliocentric motion of Venus, Tables of heliocentric motion of Mars*; Astronomical Papers of the American Ephemeris, vol. VI, part 1–4; Washington (1898).

Аналитическое разложение в ряд для движения внутренних планет.

[31] S. Newcomb; *Tables of the heliocentric motion of Uranus, Tables of heliocentric motion of Neptune*; Astronomical Papers of the American Ephemeris, vol. VII, part 3–4; Washington (1898).

Разложение в ряд координат Урана и Нептуна.

[32] F. E. Ross; New Elements of Mars; Astronomical Papers of the American Ephemeris, vol. IX, part 2; Washington (1917).

Уточнение элементов орбиты Марса из таблицы Ньюкома.

Глава 7

[33] M. E. Davies, V. K. Abalakin, M. Bursa, J. H. Lieske, B. Morando, D. Morrison, P. K. Seidelmann, A. T. Sinclair, B. Yallop, Y. S. Tjuflin; *Report of the IAU/IAG/COSPAR Working Group on Cartographic Coordinates and Rotational Elements of the Planets and Satellites: 1994*; Celestial Mechanics and Dynamical Astronomy, vol. 63, pp. 127–148 (1996).

Описание элементов вращения больших планет, используемых в настоящее время в *Astronomical Almanac* [1]. Новые результаты, получаемые рабочей группой Международного астрономического союза по картографическим координатам, публикуются приблизительно каждые три года. (Celestial Mechanics, vol. 22, pp. 205–230 (1980), vol. 29, pp. 309–321 (1983), vol. 39, pp. 103–113 (1986), vol. 46, pp. 187–204 (1989), vol. 53, pp. 377–397 (1992)).

Подробное описание вычисления физических эфемерид дано также в *Explanatory Supplement* [4].

[34] E. W. Brown; *An introductory treatise on the Lunar Theory*; Cambridge University Press (1896), Dover Publications (1960).

Описание различных методов теории возмущений для аналитического представления движения Луны.

[35] M. Chapront-Touzé, J. Chapront; *ELP 2000-85: a semi-analytical lunar ephemeris adequate for historical times*; Astronomy and Astrophysics, vol. 190, p. 342 (1988).

Средние элементы орбиты и возмущения для описания лунной орбиты на больших промежутках времени.

[36] M. C. Gutzwiller, D. S. Schmidt; *The motion of the moon as computed by the method of Hill, Brown and Eckert*; Astronomical Papers of the American Ephemeris, vol. XXIII, part 1; Washington (1986).

Аналитическое разложение в ряд для основных задач, касающихся движения Луны (без учета планетных возмущений).

[37] *Improved Lunar Ephemeris 1952–1959*; Nautical Almanac Office; Washington, 1954.

Уточненная в 1954 году версия теории движения Луны Брауна, приведены все необходимые возмущения и средние аргументы.

Хорошее введение в теорию движения Луны и основных возмущений можно найти в томе II Himmelsmechanik Буцериуса и Шнайдера [2]. Метод аппроксимирования функции с помощью полиномов Чебышева объясняется в учебниках по численным методам в математике (см. [11] и [13]).

Глава 9

[38] F. Espanak; *Fifty Year Canon of Solar Eclipses*; NASA Reference Publication 1178 Revised; NASA (1987).

Элементы и карты солнечных затмений в XX–XXI вв. с подробными сведениями о затмениях в период с 1986 по 2035 гг.

[39] H. Mucke, J. Meeus; *Canon of Solar Eclipses, –2003 to 2526*; Astronomishes Büro, Hasenwartgasse 32, Wien (1984).

Элементы 10 774 солнечных затмений для периода примерно в 4500 лет, полученные на основе теории солнечной орбиты Ньюкомба и теории движения Луны Брауна. Приведены приблизительные положения линии центрального затмения.

[40] T. Oppolzer; *Canon der Finsternisse*; Denkschriften der Math.-Naturw. Classe der Kaiserlichen Akademie der Wissenschaften Bd. 52, Wien (1883); Dover Publications, Inc., New York (1962).

Элементы солнечных и лунных затмений на период с 1207 по 2163 гг. Приведены карты линий центрального затмения.

В книге Ж. Меёса *Astronomical Algorithms* [8] описаны различные методы быстрого вычисления дат возможных затмений. Готовые таблицы для всех видов солнечных затмений приведены в [9]. Классический метод вычисления затмений с использованием Бесселевых элементов описан в *Explanatory Supplement* [4] и в книге Мюэллера [10]. Текущие предсказания солнечных затмений представлены на web-сайте Фреда Эспанака http://planets.gsfc.nasa.gov/eclipse/eclipse.html.

Глава 10

[41] J. Robertson; *Catalog of 3539 Zodiacal Stars for the Equinox 1950.0*; Astronomical Papers of the American Ephemeris, vol. X, part 2; Washington (1917).

Справочный каталог для предвычисления покрытий звезд. Содержит координаты звезд вблизи эклиптики, которые могут быть покрыты Луной, вплоть до девятой величины.

Предвычисление покрытий звезд описано во многих учебниках по сферической астрономии (например, Грина [5]) и в *Explanatory Supplement* [4]. Подробное описание исследований лунной орбиты и вращения Земли дано в *Spherical and practical astronomy* [10]. Исправленные координаты можно найти также в каталоге PPM [47].

Глава 11

[42] C. F. Gauss; *Theory of the Moon of the Heavenly Bodies Moving about the Sun in Conic Sections*; Dover Publications; New York.

Английский перевод работы Гаусса *Theoria motus corporum coelestium*, в которой он описал свой метод вычисления орбит. Книга представляет в основном исторический интерес и не годится в качестве учебника.

[43] H. Bucerius; *Bahnbestimmung als Randwertproblem I–V*; Astronomishe Nachrichten, vol. 278, 280–282 (1950–1955).

Серия из пяти статей, в которых среди прочего исследуются упрощенный и полный методы вычисления орбит Гаусса. Численные примеры иллюстрируют возможные проблемы, возникающие при вычислении орбит.

[44] D. L. Boulet; *Methods of orbit determination for the micro computer*; Willmann-Bell; Richmond, Virginia (1991).

Принципы и методы вычисления эфемерид и орбит, включая отдельные возмущения, а также методы вычисления орбит Лапласа, Ольберса и Гаусса. Приведенные программы на Бейсике доступны также на гибком диске.

Новейшее введение в проблему вычисления орбит дано также в соответствующей главе *Himmelsmechanik I* Буцериуса и Шнайдера [2].

Глава 12

[45] *SAO Star Catalog*; Smitsonian Astrophysical Observatory; Cambridge, Massachusetts (1966).

Четырехтомный звездный каталог, содержащий координаты, собственные движения и величины 258 997 звезд для эпохи 1950.0. Точность координат составляет примерно 1″.

[46] W. Dieckvoss; *AGK3 Star Catalogue of Positions and Proper Motions North of −2°.5* Declination; Hamburg-Bergedorf (1975).

Исчерпывающий каталог звезд северной полусферы для астрометрических работ. Основан на фотографических обзорах, выполненных в Боннской и Гамбург-Бергедорфской обсерваториях. Эпоха 1950.0.

[47] S. Röser, U. Bastian; *Catalogue of Position and Proper Motions*; Astron. Astrophys. Suppl. Ser. 74, 449 (1988).

S. Röser, U. Bastian; *PPM Star Catalogue*; Spectrum Akademisher Verlag, Heidelberg (1991).

S. Röser, U. Bastian; *Catalogue of Position and Proper Motions — South*; Astronomishes Rechen-Institut, Heidelberg (1993).

U. Bastian, S. Röser; *The Bright Stars Supplement to the PPM and PPM South Catalogue, Revised Edition*; Astronomishes Rechen-Institut, Heidelberg (1993).

S. Röser, U. Bastian, A. V. Kuzmin; *The 90000 Stars Supplement to the PPM Star Catalogue*; Astron. Astrophys. Suppl. Ser. 105, 301 (1994).

Преемник каталога SAO, содержащий координаты и собственные движения примерно 470 000 звезд северной и южной полусфер, отнесенные к эпохе J2000 (FK5). Различные части каталога можно найти по адресам Центра астрономических данных НАСА:

http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs /1/1146/

http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1193/

http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1206/

http://adc.gsfc.nasa.gov/adc-cgi/cat.pl?/catalogs/1/1208/

Печатная версия опубликована Spectrum Akademie Verlag, Heidelberg.

[48] W. Tirion, B. Rappaport, G. Lovi; Uranometria 2000.0; Willmann-Bell; Richmond, Virginia (1987).

Атлас северного и южного неба, содержащий 473 карты в масштабе 1,85 см = 1° для эпохи 2000. Приведены звезды до $9^{m}_{.}5$ по Боннскому обозрению, южному Боннскому обозрению и Кордовскому обозрению.

Вычисление стандартных координат описано во многих учебниках по сферической астрономии (например, [5]). Подробности см. в литературе по численным методам в математике, например, [11] и [13].

Алфавитный указатель

A

Accumulate, 274
Ada, 19
AddN, 166, 167
AddSol, 167
AddThe, 165, 168
Apparent, 215

B—C

Bessel, 209
C++, 19, 166, 240
  GNU, 181
  стандартная библиотека, 22
Cheb3D, 178, 196
  конструктор, 179
Coco, 22
Conjunct, 219, 226
Contact, 226
Contacts, 209

D—E

DMS, 23
Ecl2EquMatrix, 175
Eclipse, 197
EclTimer, 208, 209
ETminUT, 193, 223

F

FindEta, 240
Fit метод, 179
Fortran, 19
Foto, 275
Foto.dat, 275

G

Gauss, 255
Gauss.dat, 256, 263
GaussMethod, 256
GMST, 193

I—L

Intersect, 198
Java, 19
LMST, 222

M

MiniMoon, 163
MoonEqu, 174, 175, 178, 196, 197
MoonPos, 164, 168, 169, 174, 175, 178

N—O

NutMatrix, 175
Observer, 209
Occult, 211, 225
Occult.dat, 225
Orkisz.dat, 264

P

Pascal, 19
Pegasus, 217
Pegasus метод, 187, 209, 217
Pert, 166—169
Phases, 187
PhasesFunc, 185
PosAngles, 209
PPM.bin, 282
PPM.dat, 281
PPMcat, 282, 283

Q—R

Quad, 209, 223
RA_Diff, 219
ReadCatalogue, 215
regula falsi, 217

S

SAO каталог, 212
semi-latus rectum, 238
ShadowDist, 209
Site, 222
Solve, 274
SolveGL, 253, 256
solverLSQ, 272, 273
Star, 215, 225
Start, 256
SunEqu, 197

T—Z

Term, 166, 167, 170
Vec3D, 178
ZC.dat, 235

А

аберрация, 213
    света, 132, 213
абсолютная звездная величина планеты, 154
Адамса—Башфорта
    уравнение, 104
азимут, 49
алгоритм
    Кленшоу, 180
аномалия
    истинная, 73, 107, 238, 243
    средняя, 107, 244
        Луны, 162
        Солнца, 162
        суточное изменение, 244
    эксцентрическая, 75, 244
аппроксимация
    Хансена, 240
аргумент
    широты, 163
    долготы, 86, 107
    перигелия, 85, 107
аргументы
    теории движения Луны
        основные, 161
        средние, 164, 165
астероид Церера, 114
астероиды, база данных, 116
астрометрические координаты, 90, 131
астрометрия
    оценка ошибок, 276
астрономическая единица, 22, 76
астрономический ежегодник, 16
астрономический календарь, 16

Б

барицентр, 125
барицентр системы Земля—Луна, 42
Баркера уравнение, 77
бесселев год, 35
большая полуось, 74
Браге, Тихо, 161
Браун, 168
    теория, 164
Буцериус, 237

В

вариация, 161, 162
вековые изменения, 124
вектор, класс, 26
вектор состояния, 102
векторное произведение, 247
векторно-матричные операции, 27
векторы Гаусса, 87, 241
весеннего равноденствия
    точка, 33
видимая звездная величина планеты, 154
видимые координаты, 134
видимые положения, 212
возмущений теория, 96
возмущения, 96, 109, 161, 164, 167, 168, 175
    периодические
        орбиты Луны, 161
    члены, 165
возмущения от планет, 168
восход, 60
вращение
    линии узлов, 161
    планет-гигантов, 146
время, 55
    Всемирное, 56, 193, 222
    всемирное координатное, 56
    гринвичское звездное, 54, 56
    динамическое, 55
    звездное, 53
    относительность, 55
    поясное, 57
    разность ET—UT, 198
    солнечное, 56
    формат вывода, 31
    эфемеридное, 55, 129, 193, 222
Всемирное время, 56, 193
высота светила, 49
вычисление орбит
    Гаусса метод, 237
        полный, 251
        упрощенный, 246
    Лапласа метод, 237
Вычисление орбит, 237

Г

Гаусс К. Ф., 237, 252
Гаусса
    векторы, 87
Гаусса—Лагранжа уравнение, 252
Гауссов вектор, 241
гелиографическая долгота, 146, 156
гелиоцентрические координаты, 22
географическая широта, 145
геометрические координаты, 131
геоцентрическая широта, 145
геоцентрические координаты, 22
Гивенса поворот, 272
гиперболические функции, 77
главная плоскость, 190, 206
    координаты, 207

год
  бесселев, 35
  юлианский, 30
горизонтальная система координат, 49
гравитационная постоянная, 76
  гауссова, 77
Гринвичский меридиан, 193
Гринвичское звездное время, 54, 56

Д

двух тел задача, 238
диаметр видимого диска, 155
динамическое время, 55
дифференциальные
  коэффициенты, 211, 223, 224–226
долгота
  гелиографическая, 146, 156
  географическая, 191, 193, 224
  планетографическая, 146
  планетоцентрическая, 145
  эклиптическая, 33
долгота восходящего узла, 85, 107
дробная часть, 23

Е—З

ежегодник астрономический, 16
задача двух тел, 90, 96
закон
  всемирного тяготения, 72, 97
  Кеплера, второй, 72, 238
  Кеплера, первый, 72
  Кеплера, третий, 76, 244
  равных площадей, 238
законы Кеплера, 72
затмение
  кольцевое, 186, 188, 197, 206
  линия центрального затмения, 196
  моменты контактов, 209
  наибольшая фаза, 206, 208
  обстоятельства, 206
  позиционный угол, 206, 208
  продолжительность, 195
  солнечное, 16, 185, 211
  центральное, 187
  частное, 186, 187
заход, 60
звезда незаходящая, 60
звездная величина планеты
  абсолютная, 154
  видимая, 154
звездное время, 53, 193, 207, 222
  Гринвичское, 54, 56
  местное, 54
звездные сутки, 54
Земля
  ось, 34
  сжатие, 191, 192, 221
зенит, 49
Зодиакальный каталог, 212

И

Институт прикладной астрономии, 114
интегрирование численное, 102
  многошаговый метод, 103
  уравнений движения, 96
интервал, 243, 248
интерполяция, 103
  квадратичная, 62, 223
  Лагранжа, 175, 177
истинная аномалия, 73, 107
истинный полюс, 134

К

календарь
  астрономический, 16
  григорианский, 30
  реформа, 30
  юлианский, 29
камера
  оптическая ось, 267
каталог PPM, 212, 235, 266, 275, 281
каталог зодиакальных звезд, 212
квадратичная интерполяция, 62
Кеплера
  второй закон, 72, 238
  первый закон, 72
  третий закон, 76, 244
  уравнение, 76, 120, 244
кеплеровская орбита, 96
класс
  Angle, 24
  Mat3D, 27
  Pert, 126
  Polar, 26
  SolverDE, 105
  Vec3D, 26
Кленшоу, 180
комета
  Бредфилда, 266
  Вольфа, 96
  Галлея, 94
  Понс—Виннека, 96
конического сечения
  уравнение, 120, 243, 244
коническое сечение, 73
контакта моменты, 209

координаты
  астрометрические, 90, 131, 136
  в главной плоскости, 220
  видимые, 134, 136
  гелиоцентрические, 22
  геометрические, 131
  геоцентрические, 22
  декартовы, 26, 213, 222, 224
  истинные, 213
  перифокальные, 85
  планетографические
    система I, 146
    система II, 146
    система III, 146
  планетоцентрические
    Земли, 149
    Солнца, 150
  полярные, 26
  прямоугольные, 33
  средние, 213
  стандартные, 270, 275, 276
  сферические, 237, 268
  топоцентрические, 58
  экваториальные, 31, 256, 270, 276
  эклиптические, 22, 31
косвенное пертурбационное ускорение, 98
косинус суммы, 125
коэффициенты
  дифференциальные, 211, 223–226

Л

Лаборатория реактивного движения НАСА, 100
линия апсид, 73
линия узлов, 161, 163
Лоувелловская обсерватория, 116
Луна
  аномалия средняя, 162
  возмущения от планет, 168
  возмущения от Солнца, 161
  восходящий узел, 161, 162
  горизонтальный параллакс, 169, 181
  экваториальные координаты, 181
  Кеплерова орбита, 160
  наклонение орбиты, 160, 161, 163, 187, 219
  орбита, 160, 161, 211
  прецессия, 162
  приближенные координаты, 52
  притяжение Земли, 160
  притяжение Солнца, 160
  средняя долгота, 162
  средняя элонгация, 162
  тень, 211
  теория Брауна, 164
Луна (*продолжение*)
  экваториальные координаты, 174
  эклиптическая долгота, 162–164
  эклиптическая широта, 163–164

М

Майер Тобиас, 161
  лунные таблицы, 161
МАС, 72, 141
масса планет, 98
матрица
  нутации, 135
  элементарного поворота, 28
матрицы
  класс, 27
Международный астрономический союз, 72, 141
меридиан, 270
  Гринвичский, 193
  начальный, 146
    ориентация, 146
  центральный, 146
Месяц
  аномалистический, 160
  синодический, 161
метод
  DE, 104
  Ньютона, 78
  Штумпфа, 82, 90
модуль
  APC_Const, 22
  APC_Math, 23
  APC_VecMat3D, 25
  ATC_Time, 31

Н

наименьших квадратов метод, 271
наклон эклиптики, 32
наклонение, 85, 107, 241, 242
направление вращения, 141
начальный меридиан, 146
  ориентация, 146
небесная сфера
  видимое движение, 49
неизменная плоскость Солнечной системы, 141
неравенство годичное, 161, 162
новолуние, 161, 185
  дата, 185, 197, 203
нутация, 134, 197, 212
Ньютон Исаак, 161, 175
Ньютона метод, 78, 175

О

обсерватория
    Лоувелловская, 116
обстоятельства затмения, 206
общая теория возмущений, 96
объектив, 266, 267
операции
    векторно-матричные, 27
    векторные, 28
    матричные, 28
операция
    вывода, 24
    сдвига, 20
опорные звезды, 270
орбита
    кеплеровская, 96
    Луны, 160
    оскулирующая, 106
    Юпитера, периодические члены, 124
орбитальный параметр, 73
орбиты
    комет, 72
    планет, 118
оси
    планеты, позиционный угол, 142
оскулирующие элементы орбиты, 106, 109
остатки, 271
ось
    Земли, 34
    планеты, ориентация, 142
оценка ET—UT из наблюдений, 235

П

параллакс, 58
перегрузка операций, 20
перигелий, 73
    аргумент, 85, 243
    время прохождения, 244
    долгота, 243
    расстояние, 244
перигелийное расстояние, 74
Период обращения, 244
перифокальная система координат, 85
планетографическая
    долгота, 146
    широта, 145
планетоцентрическая
    долгота, 145
    широта, 144
планетоцентрические координаты
    Земли, 149
    Солнца, 150
Плеяды, 212, 225, 234
плоскости орбиты уравнение, 248
плоскость
    Солнечной системы, 141
    орбиты, 72
    пластинки, 267, 270
поворот
    элементарный, 28
позиционный угол, 206, 208, 223, 224, 226
    оси вращения, 142
    Солнца, 153
покрытие
    звезды Луной, 17
покрытия звезд, 211
Полином Чебышева, 176, 179–180
полной фазы продолжительность, 195–196, 206
полуось большая, 238, 251
полутень
    диаметр, 188, 189, 191
полюс Земли
    истинный, 134
    средний, 134
полюс мира
    северный, 49
поправка
    световая, 132
постоянная
    гравитационная, 76
постоянные
    математические, 22
    физические, 22
постоянные пластинки, 270
преобразование координат
    векторно-матричная запись, 32
    гелиоцентрических
      в геоцентрические, 39
    горизонтальные в экваториальные, 50
    между эпохами, 35
    перифокальные в эклиптические, 85
    экваториальных в горизонтальные, 50
    экваториальные в эклиптические, 33
    эклиптических в экваториальные, 33
прецессия, 22, 34, 212
приближение
    Хансена, 240
программа
    AOEcat, 117
    Coco, 22, 42
    Comet, 72, 90
    Numint, 108
    Phys, 155
    Planpos, 136
    Planrise, 70
    Sunset, 63

прямое
восхождение, 32, 50, 192–193, 211, 213
пертурбационное ускорение, 98
прямоугольные координаты, 33

Р

равноденствие, 225
весеннее, 212
расстояние, 246, 247
редукция к эклиптике, 121
рефракция, 59
решение, соответствующее орбите Земли, 250
ряд
Тейлора, 120
Фурье, 119
ряды
численное суммирование, 125

С

свет
время распространения, 186, 197, 254
скорость, 213, 255
световая
задержка, 89, 156
поправка, 132, 156
северный полюс мира, 49
сектора площадь, 238, 248
сектора и треугольника площадей
отношение, 237
секущих метод, 240
сжатие
Земли, 191, 192, 221
синус суммы, 125
система координат
горизонтальная, 49
системы счета времени, 55
склонение, 32, 50, 192
скорость
освобождения, 107
света, 22, 89, 132
Смитсониановская астрофизическая
обсерватория, 114
собственное движение, 212
соединения итерационное
вычисление, 211, 216
солнечное затмение, 16
Солнце
видимое положение, 197
приближенные координаты, 52
сплюснутость, 145, 155
средний полюс, 134
средняя аномалия, 107
стандартная библиотека языка C++, 22
сумерки
астрономические, 60
гражданские, 60
навигационные, 60
сутки
звездные, 54
средние солнечные, 54

Т

Тейлора ряд, 120
тень, 185, 211
диаметр, 188, 189, 191, 195, 196, 207
теорема
о косинусе суммы, 125
о синусе суммы, 125
теория
возмущений, 96
Теория движения Луны, 161, 163, 164, 168
средние аргументы, 164
тип
index, 26
PlanetType, 130
pol_index, 26
TimeFormat, 31
топоцентрические координаты, 58
точка весеннего равноденствия, 33
точность
вычислений, 17
машинных вычислений, 79
треугольник
площадь, 238
треугольник, площадь, 237, 243

У

углы
десятичное представление, 23
преобразование, 23
традиционное представление, 23
формат вывода, 24
узел
долгота, 242
линия узлов, 161
уравнение
Адамса–Башфорта, 104
Баркера, 77
Кеплера, 76, 120, 244
Кеплера, решение, 78
конического сечения, 120, 243, 244
плоскости орбиты, 248
центра, 119, 161
уравнение движения
численное интегрирование, 96
уравнения движения, 97

ускорение
косвенное пертурбационное, 98
прямое пертурбационное, 98
Уточненные лунные эфемериды, 172, 174

Ф

фаза, 152
фазовый угол, 151
физические эфемериды, 141
фокальный параметр, 238, 243
фокусное расстояние, 266, 267
фотография, 266
функции
гиперболические, 77
функция
Accel, 98
AddThe, 125
Bright, 154
CalcPolarAngles, 26
CalDat, 30
Ddd, 23
DMS, 23
EccAnom, 79
Ecl2EquMatrix, 34
Elements, 108
Ellip, 80
Equ2EclMatrix, 34
Equ2Hor, 51
Frac, 23
GaussVec, 88
GMST, 54
Hor2Equ, 51
HypAnom, 80
Hyperb, 81
Illum, 152
JupiterPos, 127
JupPosition, 124
Kepler, 88
KepPosition, 101
KepVelocity, 133
MiniMoon, 52
MiniSun, 52
Mjd, 29
Modulo, 23
NutMatrix, 135
Orient, 147
Parab, 84
PertPosition, 129
PlutoPos, 129, 136
PosAng, 143, 153
PrecMatrix_Ecl, 38
PrecMatrix_Equ, 38
Quad, 63
Rotation, 150
функция *(продолжение)*
Shape, 145
State, 100, 134
Stumpff, 84
SunPos, 40
гиперболическая, 240
тригонометрическая, 163, 165

Х—Ц

Хансена приближение, 240
целая часть, 23
центр малых планет, 116
центра уравнение, 161
центральный меридиан, 146
Церера, 114

Ч

часовой угол, 50, 53
часовые пояса, 57
Чебышева
коэффициенты, 179
полином, 176, 180–181
корни, 177, 180
коэффициенты, 179, 180
рекурсия, 180
приближение, 175, 177, 178

Ш

Шарлье граничная линия, 251
широта
аргумент, 242
географическая, 145, 191–192, 221, 224
геоцентрическая, 145, 192, 207
планетографическая, 145
планетоцентрическая, 144
эклиптическая, 33
Штумпфа
метод, 82, 90

Э

эвекция, 161
экватор
небесный, 212
экваториальные координаты, 31
эклиптика, 33–34, 87, 160–161, 185, 211, 216, 241
изменение наклона, 32
наклон, 32
эклиптическая
долгота, 33
широта, 33
эклиптические координаты, 22, 31

эксцентриситет, 73, 238, 243
эксцентрическая аномалия, 75
элементарный поворот, 28
элементы орбиты, 72, 237, 241, 249
    вековые изменения, 124
    оскулирующие, 106
    планеты, 100
эллипс, 244
эллипсоид вращения, 191, 221
элонгация, 152
Энке, 251
эпоха, 32, 212, 225
    B1950, 35
    J2000, 35
    текущей даты, 35
эфемеридное время (ET), 55, 129, 193, 222
эфемериды, 109
    высокоточные, 100
    лунные, 181
эфемериды *(продолжение)*
    уточненные, 172
    физические, 141

Ю

юлианский
    год, 30
    день, 29
    день, модифицированный, 29
    календарь, 29, 30
    период, 29

Я

язык программирования
    Ada, 19
    C++, 19, 105
    Fortran, 19
    Java, 19
    Pascal, 19